Master's thesis

Master's Programme in Computer Science

Real-time satellite data processing platform architecture

Otto Hyytiälä

June 4, 2021

Faculty of Science

University of Helsinki

Supervisor(s)

Assoc. Prof. Laura Ruotsalainen, Dr. Lauri Häme

Examiner(s)

Assoc. Prof. Laura Ruotsalainen, Dr. Lauri Häme

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki,Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Study programme
Faculty of Science	Master's Programme in Computer Science

Tekijä — Författare — Author

Otto Hyytiälä

Työn nimi — Arbetets titel — Title

Real-time satellite data processing platform architecture

Ohjaajat — Handledare — Supervisors

Assoc. Prof. Laura Ruotsalainen, Dr. Lauri Häme

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master's thesis	June 4, 2021	50 pages

Tiivistelmä — Referat — Abstract

Remote sensing satellites produce massive amounts of data of the earth every day. This earth observation data can be used to solve real world problems in many different fields. Finnish space data company Terramonitor has been using satellite data to produce new information for its customers. The Process for producing valuable information includes finding raw data, analysing it and visualizing it according to the client's needs. This process contains a significant amount of manual work that is done at local workstations. Because satellite data can quickly become very big, it is not efficient to use unscalable processes that require lot of waiting time. This thesis is trying to solve the problem by introducing an architecture for cloud based real-time processing platform that allows satellite image analysis to be done in cloud environment. The architectural model is built using microservice patterns to ensure that the solution is scalable to match the changing demand.

ACM Computing Classification System (CCS)
Information systems → Information systems applications → Spatial-temporal systems → Geographic information systems

Avainsanat — Nyckelord — Keywords

architecture, satellite data, GIS, spatial data, mircoservices

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Software Systems specialisation line

Acknowledgement

I would first like to thank my supervisors Associate Professor Laura Ruotsalainen and Doctor Lauri Häme for continuous support and guidance for this thesis.

I would also like to thank all my colleagues at Terramonitor for the helpful support and valuable comments.

Finally, I would like to acknowledge my family and friends for the support through the process of writing this thesis.

Helsinki, 4th of June 2021
Otto Hyytiälä

Contents

Abbreviations

API	Application programming interface
AWS	Amazon Web Services
COG	Cloud Optimized GeoTIFF
CRS	coordinate reference system
DBMS	database management system
DDD	Domain-Driven Design
EO	Earth Observation
ESA	European Space Agency
GeoTIFF	Geographic Tagged Image File Format
GIS	Geographic Information System
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a service
NDVI	Normalized Difference Vegetation Index
NIR	Near-infrared
NLS	National land survey of Finland
OGC	Open Geospatial Consortium
PaaS	Platform as a service
RPP	Real-time prosessing platform
S3	Simple Storage Service in AWS
SaaS	Software as a service
SAR	Synthetic aperture radar
SOA	Service-oriented architecture
SQL	Structured Query Language
TIFF	Tagged Image File Format
WCPS	Web Coverage Processing Service
WCS	Web Coverage Service
WKSS	well-known scale set
WMTS	Web Map Tile Service

1 Introduction

Satellites orbiting the earth are producing images that can be used in many professional fields. Data produced by satellite instruments can be used to perform numerous analyses on any geographical area of interest. By performing analysis to satellite data, the data can become valuable information for organizations in different fields.

Satellite imagery solutions can solve problems that include observing something in the real world. These solutions can for example be the detection of illegal loggings or forest fires. By further processing satellite data with analysis, elements that are not visible for the human eye can be exposed. For example, infrared light can be captured by satellites but is not visible to the human eye. By processing infrared and visible light data, valuable new data can be created.

Finnish space data company Terramonitor uses optical satellite images to provide services for customers in different fields (Terramonitor, 2021). Terramonitor performs analysis on the given area of interest to obtain valuable information that can be delivered to the customer depending on their needs. For example, in the forestry sector, the customer can be interested in the condition of their forest. In that kind of task, Terramonitor would find the suitable satellite images for the customer's area and perform analysis for the raw satellite data to generate results about the condition of the forest.

Datasets containing satellite imagery can quickly become large because raw data contains a large amount of information for the given area. Performing analyses locally can be inefficient since high-speed data transfer capability and powerful hardware is needed to process satellite data.

Terramonitor has developed a method to pre-process satellite data efficiently. In pre-processing, a set of satellite images for the same area is used and cloud-free parts of the images are selected. Images then go through an atmospheric correction, that reduces the effect of the atmosphere on the satellite images. The pre-processed satellite data can be used to provide solutions to different clients. The scope of this work is related to the process where pre-processed data is further processed and analysed. The process also includes the creation of the visualization, if the results can be visualized as an interactive map. The process from pre-processed data to result visualization can be done manually and offline, but it requires more work before results can be delivered. By automating

the process in a cloud environment, time is saved, and the efficiency of result delivery is increased.

Currently performing analysis requires multiple steps related to the data transfer and visualization. Those steps require extra attention from the specialists trying to generate new information from satellite data. A new platform is needed to give specialists a way to interact with the satellite data without having to manually handle the data transfer and visualization.

The objective of this thesis is to study new methods for performing analysis on satellite images more efficiently. By obtaining information about data storage, transfer and visualization of satellite imagery an architecture for a real-time processing platform is introduced. The real-time processing platform will have the capability to run analyses for any area of interest and get the result quickly by automating the extra steps that are now needed to produce visualizations from the results. This platform increases the efficiency of satellite data analyses by offering a cloud-based solution that has an architecture built for handling large amounts of satellite data.

Three research questions are formed to help this thesis to proceed in the creation of the architecture. This thesis is guided by the following research questions:

> ***RQ1:*** *How is different kind of spatial data stored and hosted?*
>
> ***RQ2:*** *What are the use cases for the real-time processing platform?*
>
> ***RQ3:*** *What architectural decisions need to be made for real-time processing platform?*

By answering these research questions this thesis is finding a way to build an efficient architecture for a cloud-based real-time processing platform. Being able to visualize the results of the analysis run by the platform is an important requirement for the platform. One way to visualize the results is in the form of an image. And when processing geospatial data, those images can form an interactive map that is explorable by the user.

The first research question is related to the data storage and hosting of the results in a way that the data can be visualized as a map. Storing images can increase the requirements for the hardware because the size of map data can be very large. This question addresses existing solutions to find out the best practices for storing map data, and how this data is hosted efficiently to be visualized.

The second research question considers the requirements for the real-time processing platform. Requirements can be found by observing previous and ongoing projects within Terramonitor. The focus is to find problems in the projects that could be solved more efficiently with a real-time processing platform. Use cases and requirements for the platform are determined by interviewing employees in the projects and by studying the project plans and results.

In the third question, the solutions from question one and the requirements from question two are combined to produce an architecture model for the real-time processing platform. The architectural model includes a model for storing, transferring and visualizing the data. The overall architecture of the platform integrates the architecture into one cloud-based platform. In addition to the three models mentioned above the answer to this research question includes a solution for the cloud infrastructure that is required for deploying the platform. The overall architecture provides an end-to-end solution for the Real-time Processing Platform.

Based on the research questions the structure of the thesis is as following. In section 2 the background for this work is presented by introducing the basic knowledge needed to understand the usage of geospatial and satellite data. Section 2 also introduces the concept of satellite image analysis that is conducted in Terramonitor to provide space data solutions for different fields. In section 3 the current literature is studied to find solutions and architectural models that can be used in the platform architecture. In section 4 the architecture for the real-time processing platform is created based on the answer to research question 1. Section 4 gathers the requirements for the platform and uses them to form an architectural model considering the end-to-end solution, which is presented in section 5.

2 Background

Remote sensing satellites use different instruments to capture spatial data of the earth. Spatial data or geospatial data is data that includes a location on earth. Spatial data captured by satellites can be optical images of the earth surface or for example observations of different gasses in the atmosphere. With numerous data sources and types, this spatial data can be used to produce visualizations of different objects, events or phenomena that are located on earth. If the data can be attached to a certain time it is called spatiotemporal data, because it has both spatial and temporal features. For example, a satellite image from the city of Helsinki captured on June 20th, 2020 is a spatiotemporal image.

Spatial data can be divided into two groups, vector and raster data. Vector spatial data represent objects in the world by using points, lines and areas whereas raster data have multiple cells where each cell has a location and value. Figure 2.1 shows examples of spatial data in raster and vector format. Vector data structure uses generally less storage space, because of the structure of the data. Vector data is a good option for handling individual objects from the real world, such as roads. In vector data it is possible to define the topology between objects, for example, defining which roads are connected. Raster data is a good option for describing surfaces with changing attributes. Raster data takes more storage space because every cell of the area needs to be stored individually. Raster data works directly with remote sensing images because images can be represented in raster format. Raster data can be easily used in different spatial data analysis (Kwang-Soo Kim et al., 1997).

Modern satellites capture images digitally. Each of the digitally captured images forms a two-dimensional array of pixels. The density of the pixels is presented as the spatial resolution of the satellite image. Spatial resolution tells the size of the smallest object that is resolved by the sensor (Liang et al., 2020, Chapter 1). For example, a satellite image of a city where individual cars are visible can have a spatial resolution of one meter. Figure 2.2 gives an example of how spatial resolution affects the image captured from a city.

Remote sensing satellites can capture different parts of the electromagnetic spectrum. Spectral resolution defines the different bands that are present in the satellite's images (Liang

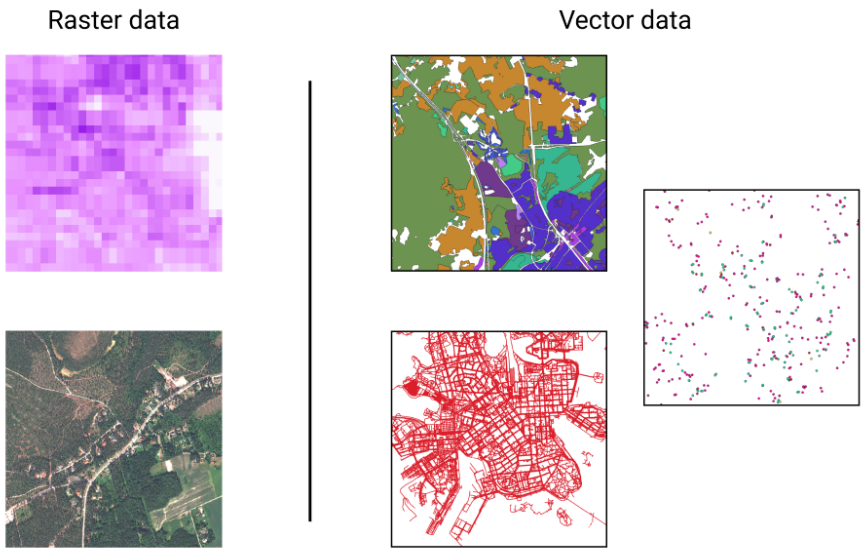

Figure 2.1: Examples of raster and vector based spatial data (Terramonitor, 2021; OpenStreetMap contributors and Geofabrik GmbH, 2021).

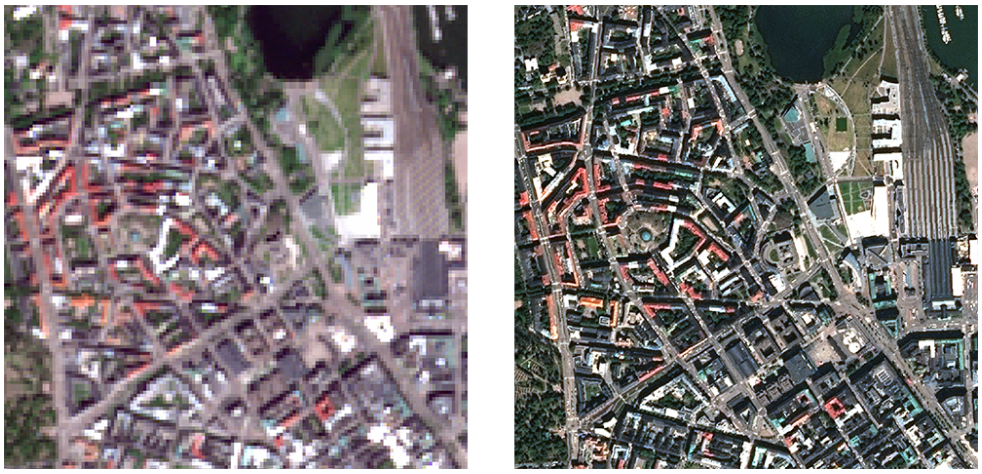

Figure 2.2: Spatial resolution of 10 meter from Sentinel-2 satellite on the left and 1.5 meters on the right from SPOT-7 satellite (Terramonitor, 2021).

et al., 2020, Chapter 1). For example, the spectral resolution of a satellite producing only true colour images is three with the bands of red, green and blue. Bands outside the visible portion of the electromagnetic spectrum can be used to perform analysis and calculating different indices from the satellite data. When visualizing the results those bands need to be moved to the visible portion to let the human eye see the result. Figure 2.3 shows an example of visualization of three bands where the infrared band is used to replace a colour channel.

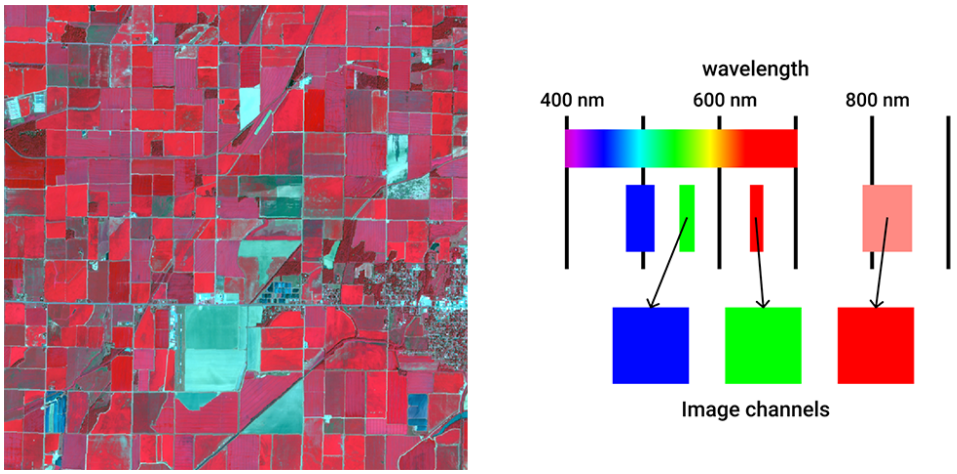

Figure 2.3: Visualization of satellite image from Sentinel-2 where red channel represents Near Infra-Red band (ESA, 2021; Terramonitor, 2021).

The radiometric resolution describes the range of values for each pixel. For example, the radiometric resolution of 8 bits means that each pixel has a value between 0 and 255 (Liang et al., 2020, Chapter 1). After pre-processing if the pre-processed images do not have values on the whole scale of available bits the image cannot be observed easily by the human eye, because it appears to be too dark. To enhance the readability of the image it can be scaled so that the pixel values are distributed more evenly on the radiometric range. By doing so the image uses more of the available values and the image is not too dark. Figure 2.4 shows pre-processed Sentinel-2 true colour and false colour images, and scaled versions of those pre-processed images.

European Space Agency (ESA) is one of many organizations that develops and maintains remote sensing satellites. Sentinel-2 is an earth observation mission by ESA, and it contains two optical remote sensing satellites. Sentinel-2 has three different spatial resolutions and its spectral resolution is 13 (ESA, 2021). Table 2.1 lists the different bands and spatial resolutions provided by Sentinel-2 satellites. The temporal resolution of Sentinel-2 is five days, which means that areas are revisited every five days (Liang et al., 2020, Chapter 1). Sentinel-2 data is free to access and the data can be used to perform analysis on different areas. Since the most accurate spatial resolution is ten meters, the satellite does not perform well when observing tiny details. The left side of Figure 2.2 shows the inaccuracy of Sentinel-2 when observing urban areas.

With a satellite image that contains multiple bands, each pixel captured by the satellite can have multiple values. The amount of values for a pixel depends on the spectral resolution of the image. Multiple combinations of the values for different bands can be used to

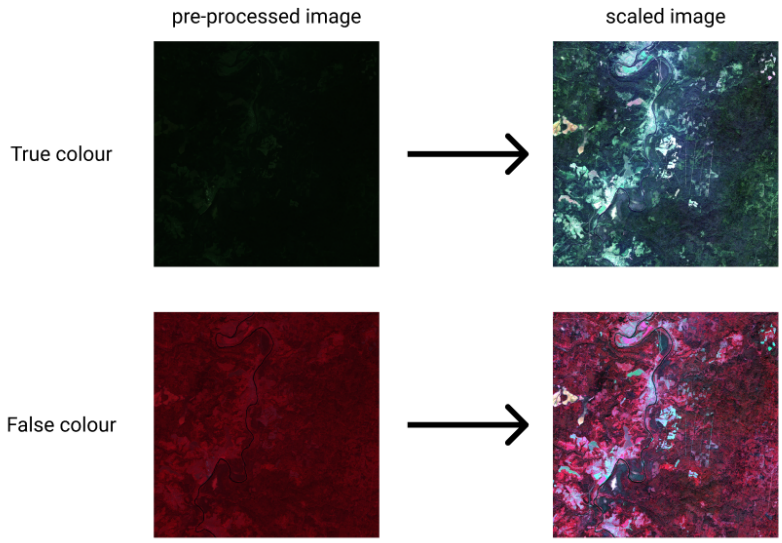

Figure 2.4: Pre-processed unscaled Sentinel-2 True and false colour images, and the scaled true colour and false colour images (Terramonitor, 2021).

Table 2.1: List of Sentinel-2 satellite bands (ESA, 2021)

band number	spatial resolution	central wavelength	description
2	10 m	490 nm	blue
3	10 m	560 nm	green
4	10 m	665 nm	red
8	10 m	842 nm	near-infrared
5	20 m	705 nm	visible and near-infrared
6	20 m	740 nm	visible and near-infrared
7	20 m	783 nm	visible and near-infrared
8b	20 m	865 nm	visible and near-infrared
11	20 m	1610 nm	short-wave infrared
12	20 m	2190 nm	short-wave infrared
1	60 m	443 nm	aerosol detection
9	60 m	945 nm	water vapour detection
10	60 m	1375 nm	cirrus detection

generate new data from the images. For example, Normalized Difference Vegetation Index (NDVI) is a well-known index that corresponds to the amount of live green vegetation in a certain area. NDVI can be calculated for a pixel with the Near Infra-Red (ρ_{nir}) and red

(ρ_{red}) bands using formula 2.1 (Liang et al., 2020, Chapter 12). Figure 2.5 visualises the red and NIR bands values on a black to white scale using Sentinel-2 satellite's bands 4 (red) and 8 (nir), and the NDVI result that is created using those values.

$$\mathrm{NDVI} = \frac{\rho_{\mathrm{nir}} - \rho_{\mathrm{red}}}{\rho_{\mathrm{nir}} + \rho_{\mathrm{red}}} \tag{2.1}$$

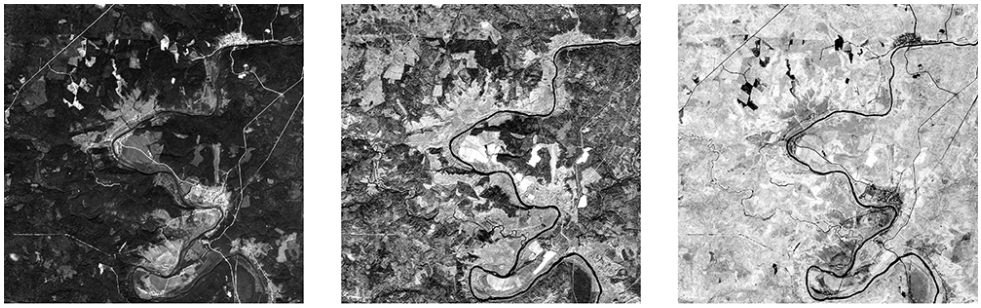

Figure 2.5: Visualization of red and near infra-red (nir) bands, and NDVI result (Terramonitor, 2021)

The calculated NDVI only has one channel that represents the NDVI value for a pixel. A typical way to visualize one channel results is to assign a colour-ramp for the values to use colours in the visualization. Depending on the data the threshold values for the colour ramp can be adjusted to bring out the key findings in the result. The coloured NDVI result can be used to find areas with different vegetation density. Figure 2.6 shows a visualization of NDVI with colour spectrum blue-green-yellow-red. In the upper left corner of the figure, there are blue areas that indicate loggings made to the forest. Compared to the grayscale visualization of NDVI in Figure 2.5, the coloured version gives a clearer view of the results.

Calculating NDVI is only one example of an analysis that can be done to satellite data. The field of Geoinformatics has been developing indexes and methods that can be applied to remote sensing data. These methods help to generate new information based on the satellite data which can be used to provide solutions to various problems. The real-time processing platform is going to provide a way to make these analyses so that result visualization is created as part of the analysis process. By doing so the analyst can focus on the data analysis because the infrastructure handles the data transfer and visualization. This way the result visualization can be delivered more effectively to the client.

This section introduced the background for this thesis. The purpose of this section was to give basic knowledge that is needed when processing and analysing spatial satellite data. Satellites used in the analyses are capable to capture a wide range of data with the devices onboard them. Optical satellites introduced in this section use optical sensors to capture

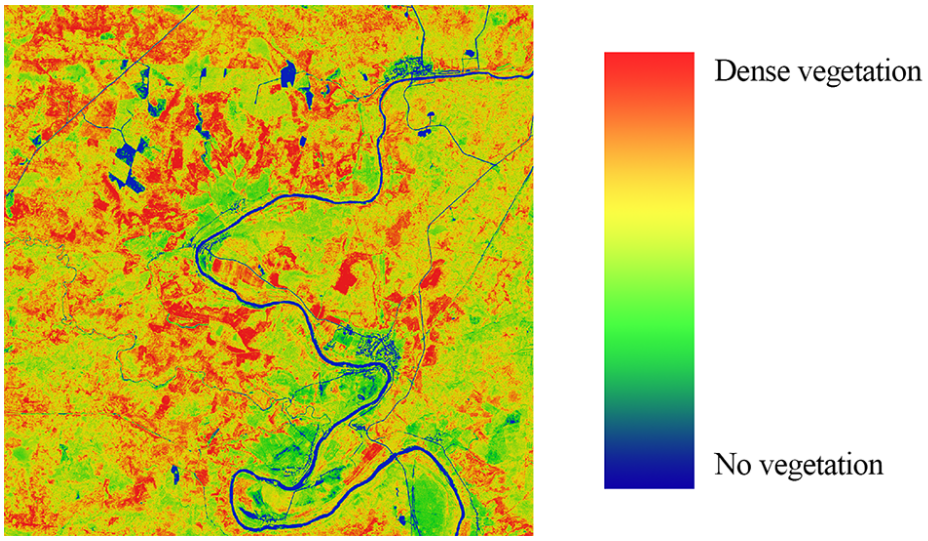

Figure 2.6: Coloured NDVI visualization and legend describing the colours used in the visualization (Terramonitor, 2021).

data from the electromagnetic spectrum. Another common technique to capture data with earth observation satellites is to use synthetic aperture radar (SAR). This section also introduced the four basic resolution terms that can be used to classify different satellites and their capabilities. With the knowledge of spatial, temporal, spectral and radiological resolution the right satellite can be chosen to suit the needs of the project.

3 Methodology of processing satellite images

The goal of this study is to find out the architectural decisions that are needed to build an effective service to run analyses for satellite images. The methodology of this thesis covers efficient ways to store, host and visualize satellite data or more generally map data. Secondly, this thesis gathers the requirements for the real-time processing platform. With the found knowledge and the set requirements, this thesis is building an architecture for a scalable and efficient platform that reduces the amount of work needed to generate solutions based on satellite data.

3.1 Research methods

The literature review is used as a research method to get an answer to research question one. Literature review gathers knowledge about the current situation of technologies that are used in Geographic Information Systems (GIS) that involve the handling of spatial data. This thesis will mainly focus on solutions that handle raster data such as satellite images rather than solutions that handle vector spatial data. Material for literature review is searched from various academic publications. The main sources for this literature review are computer science publications, but material is also searched from geographic information systems related publications. The field of Geographic Information Systems has a big need for map services and because of that GIS related publications can have valuable material. Publications that contain data about web-based GIS solutions that handle raster data are prioritized for the literature review. Prioritized publications are used to gather best practices considering data storage and hosting, data transfer techniques and used protocols. There is a broad offering of open-source tools that can be used to build GIS solutions. Material about those tools gives a good overview of some methods and techniques used in GIS solutions and material or publications involving those products are used in this literature review as well.

3.2 Open Geospatial Consortium

Open Geospatial Consortium (OGC) is a worldwide organization that has an important role in the field of earth observation (EO). OGC maintain standards that are used in various EO solutions. Standards help technologies to work more reliably together and with common standards, the efficiency of development of new solutions can be improved (Simonis, 2019). OGC offers a wide range of free and open standards that will benefit the field of earth observation and help the development of new applications.

OCG maintains standards from discovering to visualizing geospatial data. For example, there are multiple standards for storage and transfer technologies for different types of geospatial data (OGC, 2021; Trakas and McKee, 2011). Standardized methods for handling spatial data improve the usage of EO applications by taking care of certain aspects related to the architecture of the solution, like storage of spatial data or integration with other EO solutions. EO solutions, like satellite image analysis, are typically customized to fit the client's needs. OGC standards can increase the efficiency of delivering these customized EO solutions because standards allow organizations to automate certain processes (Trakas and McKee, 2011).

This thesis uses multiple standards provided by OGC to produce efficient architecture for a real-time processing platform. Using OGC standards as part of the architecture ensures that the data input and data output are implemented in a way that other external EO solutions can work more reliably with the real-time processing platform. The next sections will present OGC standards related to storage, processing and hosting of spatial data.

3.2.1 Geographic Tagged Image File Format (GeoTIFF)

Geographic Tagged Image File Format (GeoTIFF) is a standard maintained by OCG. GeoTIFF standard provides a way to store remote sensing data in one file. GeoTIFF files use encoding from Tagged Image File Format (TIFF), which is a widely used raster data image format. TIFF files have been widely used as a file format for remote sensing images, but the lack of spatial data features like location in metadata created a demand for a new file format (Ritter and Ruth, 1997).

TIFF file format supports a wide range of image sizes and because of the lossless compression of the format, TIFF images can preserve all the details from raw data (Wiggins et al., 2001). TIFF file can be also used to store multiple image layers, which makes it a

good way to store multiple bands in one file. GeoTIFF standard introduced several new tags that cover the spatial features of a remote sensing image.

To connect a remote sensing image to an actual location on earth geocoding and georeferencing is needed. Remote sensing image represents raster space and it is built with a two-dimensional grid containing pixel values. World space uses the model of earth with the three-dimensional coordinate system to locate a position on the earth. To connect two-dimensional raster space to three-dimensional world space a model space is needed in between to first convert the world space into a flat surface (Ritter and Ruth, 1997). Figure 3.1 visualizes the relationship between raster, model and world space using georeferencing and geocoding.

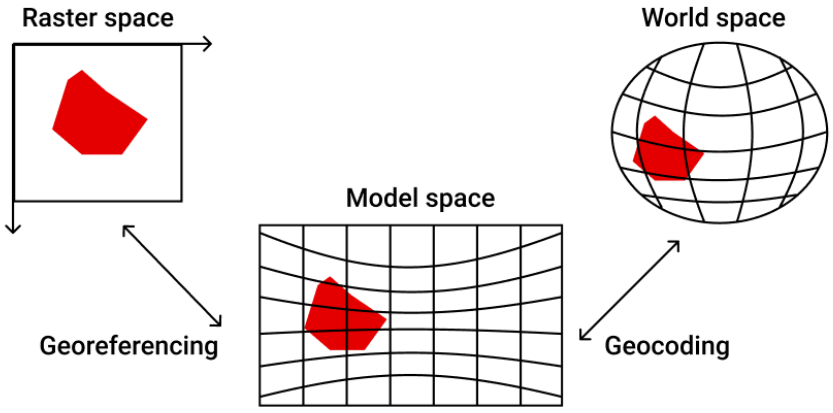

Figure 3.1: Raster, model and world space visualised (Ritter and Ruth, 1997).

GeoTIFF offers a solution to this problem by providing a way to use standardized tags in the file metadata to attach GeoTIFF image to a location in the world model. GeoTIFF format expands the traditional TIFF format by using GeoKeys in addition to TIFF tags. These GeoKeys are used to describe the geocoding of the GeoTIFF. GeoKeys contain information about the projection that is used to convert the world space to model space. These GeoKeys are written in three TIFF tags. The *GeoKeyDirectoryTag* TIFF tag is used as a directory to point out where values for different GeoKeys are located. The values of the GeoKeys are stored into *GeoDoubleParamsTag* or *GeoAsciiParamsTag* depending on the data type of the value. The rest of the tags are used to describe the georeferencing and the features of the raster and they use the traditional way to store data to TIFF tags. Georeferencing tags are used to attach the raster space to the model space (Ritter and Ruth, 1997; OGC, 2019a). Because GeoTIFF uses standardized tags in geocoding and

Table 3.1: List of GeoTIFF TIFF tags used in geocoding and georeferencing (Ritter and Ruth, 1997; OGC, 2019a)

Registry ID	Tag identifier	Description	Usage
34735	GeoKeyDirectoryTag	Reference to GeoKey values in 34736 & 34737	geocoding
34736	GeoDoubleParamsTag	Double GeoKey values for geocoding	geocoding
34737	GeoAsciiParamsTag	ASCII GeoKey values for geocoding	geocoding
33922	ModelTiepointTag	reference point between raster and model space	georeferencing
33550	ModelPixelScaleTag	Raster image scale	georeferencing
34264	ModelTranformationTag	Exact rotation of raster	georeferencing

georeferencing the format performs well in most GIS systems and with standardized file format moving files between systems is easier. Table 3.1 lists all six TIFF tags that are introduced in GeoTIFF format.

Cloud Optimized GeoTIFF (COG) is a way to store GeoTIFF which makes it more flexible to use in a file server. COG format is capable of serving only a defined range of data without the need for sending the full raster by utilizing the HTTP GET Range requests. The OCG format achieves this by organizing the pixels in the raster with tiling or overview technologies. The tiling method creates tiles of the raster for internal use to allow an easy way to request partial data. The overview method creates downsampled images of the raw data. This makes it faster to request overviews of the raster on different zoom levels because the response does not contain all the pixels from the raster but for example only a hundred of them (Holmes and Rouault, 2017; Durbin et al., 2020).

3.2.2 Web Mercator

Web Mercator or Pseudo-Mercator is a mapping projection widely used in digital mapping. It is very similar to a Mercator projection that is used in navigation and printed maps. Mercator projection is a cylindrical map projection originally developed by Gerardus Mercator in 1569 and because it preserves angles around points it is suitable for navigation. Distortion cannot be avoided in mapping because three-dimensional earth is needed to

be converted to a two-dimensional surface and in Mercator projection, the distortion increases when moving away from the equator. When converting earth into a flat surface using Mercator projection it is possible to use either a spherical or ellipsoidal model of the earth. In two-dimensional map surface coordinates of a place can be presented as x and y. If using a sphere as an earth model with a radius R, the map coordinates can be calculated using formulas 3.1 and 3.2, where λ is the latitude and φ is the longitude, both in radians. R represents the radius of the sphere and λ_0 represents the latitude of central meridian (Battersby et al., 2014).

$$x = R(\lambda - \lambda_0) \tag{3.1}$$

$$y = R \ln \tan \left(\frac{\pi}{4} + \frac{\varphi}{2} \right) \tag{3.2}$$

Coordinates can be also calculated using the ellipsoid model of the earth that is a more precise model. Coordinates calculated using ellipsoid model use formulas 3.3 and 3.4, where a is the semi-major axis of the ellipsoid and e is ellipsoid's first eccentricity that is defined using formula 3.5, where ε is the linear eccentricity defined by 3.6, where b is the semi-minor axis of the ellipsoid (Battersby et al., 2014).

$$x = a(\lambda - \lambda_0) \tag{3.3}$$

$$y = \frac{a}{2} \ln \left[\tan \left(\frac{\pi}{4} + \frac{\varphi}{2} \right) \left(\frac{1 - e \sin \varphi}{1 + e \sin \varphi} \right)^{\frac{e}{2}} \right] \tag{3.4}$$

$$e = \frac{\varepsilon}{a} \tag{3.5}$$

$$\varepsilon = \sqrt{a^2 - b^2} \tag{3.6}$$

Web Mercator is a special case of Mercator projection and it always uses the sphere model to calculate coordinates. Web Mercator uses the WGS 84 standard that defines the radius R to be 6378137 meters. Because of the use of the spherical model and not the most precise model available Web Mercator has more distortion, but on big scales, the differences between Mercator and Web Mercator are very little. Web Mercator is a good choice for GIS systems because it simplifies the Mercator projection that leads to simpler calculations (Battersby et al., 2014).

3.2.3 Web Map Tile Service (WMTS)

Internet map services are currently widely used and there are web-based GIS solutions also available. A typical use case is to look at only a small piece of the earth's surface at a time. It is not efficient to transfer areas that are not viewed to the user. For example, transferring a huge GeoTIFF file over the internet to only look at a small area in it will require a lot of bandwidth. Web Map Tile Service (WMTS) is a standard maintained by OGC to solve the problem of transferring raster data to users over the internet. With WMTS standard user can request only the areas that are viewed at the time.

WMTS protocol uses a Two-Dimensional Tile Matrix Set to divide the earth into tiles. Tile Matrix Set contains one or more Tile Sets. Each Tile Set contains Tile Rows and Tile Columns that define the tiles. When moving to the next Tile Set an individual tile is distributed to several tiles on the next Tile Set. The number of rows and columns on each Tile Set depends on the coordinate reference system (CRS) and tile scale used in the Tile Matrix Set. To overcome the problem of mixing different CRS and tile scales OGC has standardized various well-known scale sets (WKSS). Web Mercator Quad Tile Matrix Set is a WKSS that uses Pseudo-Mercator (WGS 84) as a CRS. Web Mercator Quad Tile Matrix Set uses a quadtree where each tile has four corresponding tiles on the next Tile Set (OGC, 2019b). Figure 3.2 shows the first three Tile Matrices in Web Mercator Quad TileMatrixSet.

WMTS protocol gives the user a way to request tiles from TileMatrixSets and the WMTS interface has three endpoints that can be used. Service metadata resource describes the features of the WMTS interface including the different layers available and the WKSS used in the layers. Tile resource shows the requested tile from the TileMatrixSet. The third resource called FeatureInfo provides detailed information about a specific part of the tile. The resources are linked to specific operations that are sent to the WMTS interface using HTTP protocol (OGC, 2019c).

GetCapabilities operations respond with structured Service metadata and it tells user information about the WMTS version and the service provider. The service metadata is followed by defining the contents provided by the WMTS interface. Contents have a list of layers that are available and a list of WKSS that can be used in requests. Service metadata can be imported to GIS solutions that can be used to visualize layers (OGC, 2019c).

A tile can be requested with **GetTile** operation and the user must specify first the layer,

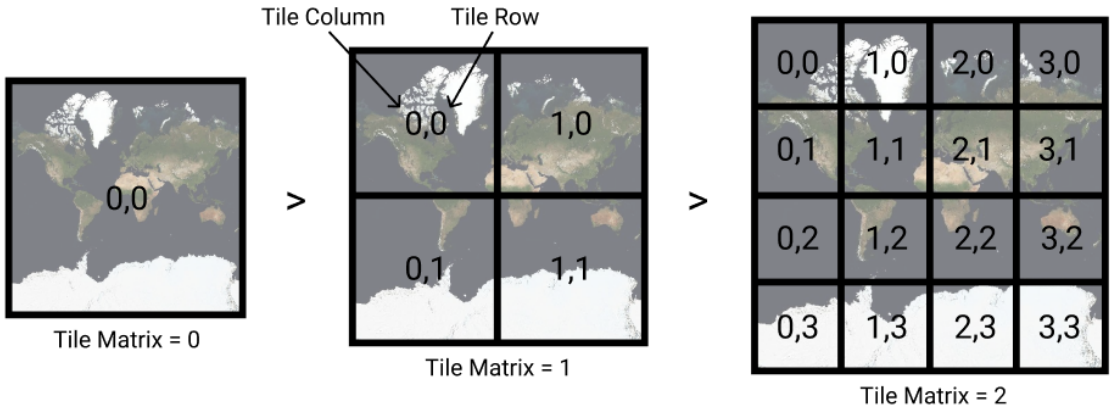

Figure 3.2: Tile Matrices 0, 1 and 2 of Satellite image world map using Web Mercator Quad TileMatrixSet (Terramonitor, 2021).

style and format being requested and which Tile Matrix Set is used. A layer can have multiple Tile Matrix Sets if there is many WKSS to choose from. To distinguish the actual tile from the Tile Matrix Set user must specify the Tile Matrix, Tile Column and Tile Row of the requested tile (OGC, 2019c). Tile Matrix can be thought of as a zoom level because when moving to a higher Tile Matrix the number of tiles is increased and so is the details in the tiles. For example, in Web Mercator Quad Tile Matrix Set the number of tiles in a specific Tile Matrix can be calculated with 4^x where $x = TileMatrix$ (OGC, 2019b). Figure 3.3 gives an example of **GetTile** request and respond using HTTP protocol.

3.2.4 Web Coverage Processing Service (WCPS)

Satellite data can be stored in an n-dimensional array where each band of the satellite image is represented by a two-dimensional array. Web Coverage Processing Service (WCPS) is a standard that is used to retrieve data from multi-dimensional spatial data. WCPS standard is maintained by OGC and it is included in OGC's coverage standards that include standards that are used to retrieve spatial coverage data like sensor or statistic data. Spatial data, for example, satellite images stored in the server are called coverages. Each coverage consists of a set of coordinates with a value. OCG's different coverage standards

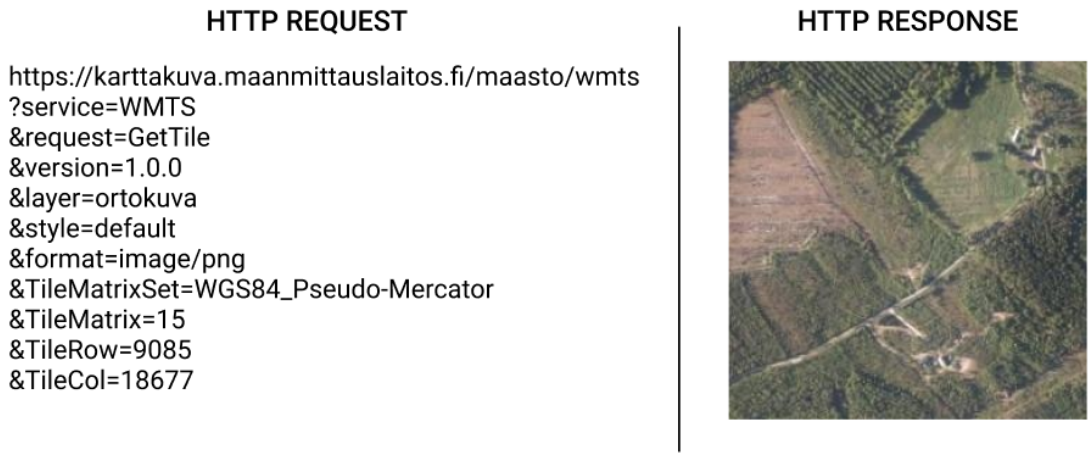

Figure 3.3: WMTS GetTile request for map tile from Finland orthophoto map (NLS, 2021).

provide functions that retrieve values based on coordinates in specific coverages (Baumann, 2010).

WCPS expands the Web Coverage Service (WCS) standard that is used to retrieve coverage information from multi-dimensional spatial data without any processing. To understand the features of WCPS it is important to know the basics of WCS. WCS **GetCapabilities** operations returns the available coverages on the server. Detailed information about the coverages can be requested with **DescribeCoverage** operation. Actual coverage data can be requested using **GetCoverage** operation where client must specify the options for the request (Aiordăchioaie and Baumann, 2010; OGC, 2018).

WCPS introduced a way to make analyses for the coverages and get the results using a specific query language. The WCPS requests can loop over the selected coverages and process the values in them to create new results. WCPS supports various processing operators including basic arithmetic operations that can be used in the requests. The return clause in WCPS requests provides supports for different file formats like image formats and tabular data formats (Baumann, 2010; OGC, 2009). For example, in the following WCPS code snippet, NDVI value between -1 and 1 is calculated for coverage SatImage. The variable s represents the coordinates in the multi-dimensional spatial data. The calculation uses only data from coordinates where the average intensity of the red band is over 127. Figure 3.4 shows the result after calculation when a white and black

image is created with the threshold of 0,6.

```
for s in ( SatImage )
where avg( s.red ) > 127
return encode(((s.nir-s.red)/(s.nir+s.red))>0.6,"png")
```

Figure 3.4: NDVI result image created with WCPS request, where NDVI values are between -1 and 1. White areas represent NDVI values greater than 0.6 (Terramonitor, 2021).

3.3 Cloud platforms

Cloud platforms provide flexible and scalable services that are meant to make managing solutions easier and more robust. Cloud platforms can be recognized for their capability to offer on-demand services from widely scalable resource pools that provide unlimited resources for computing and storage (Mell and Grance, 2011). Cloud platforms can provide software, platform, and infrastructure as a service, depending on the needs of the organization. Cloud platforms can for example provide virtual machines for infrastructure or cloud storage for big data loads. Organizations can move their infrastructure to cloud providers premises and by doing that organizations can concentrate more on their work because most of the responsibilities concerning the infrastructure are taken care of cloud platform (Xu and B. Li, 2013).

Software as a service (SaaS) is a service model that provides access to a dedicated application that is running in the cloud and all maintenance is taken care of by the cloud provider. Platform as a service (PaaS) provides a way to deploy own applications on top of the cloud infrastructure without the need for configuring the server instance or operating

systems. The environment is maintained by the cloud provider. Infrastructure as a Service (IaaS) gives the user the most control over the cloud services by providing the server and networking capabilities but leaving the server and operating system configuration for the user (Saraswat and Tripathi, 2020; Mell and Grance, 2011). Figure 3.5 shows an overview of SaaS, PaaS and IaaS service models compared to on-premises hosted services.

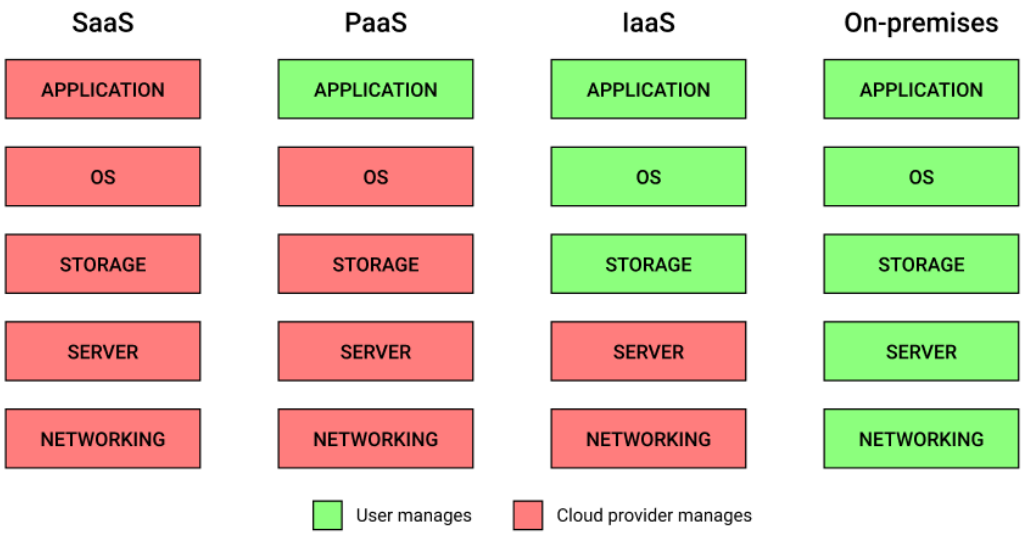

Figure 3.5: Manageability in SaaS, PaaS and IaaS compared to on-premises hosting (Mell and Grance, 2011).

Amazon Web Services (AWS) provide multiple cloud services including computing instances and data storage. Services such as AWS has made it possible to process large satellite data more efficiently (S. Li et al., 2016). The size of analysed data depends highly on the size of the area which affects the processing power needed. Because AWS and other cloud providers offer pay-per-use business models the costs of handling spatial data will vary, and it can lower the overall costs for computing infrastructure (Riisager et al., 2016).

Efficient GIS solutions can be built by using services from AWS and it is possible to connect those services within the cloud to achieve a fast connection between services, that is needed to transfer a large amount of satellite data. AWS also provides a data library that contains open spatial datasets like images from Sentinel-2 satellites (AWS, 2021).

3.4 Storing satellite data

Storage has an important role in the process of satellite image analysis. Storage and issues related to it are first introduced when new satellite data is received for processing or analysis. The raw data must be stored before it can be manipulated into the needed form. After analysis, the generated data also must be stored.

The database is a part of software that is intended to store information. One of the most common database types used in applications is a relational database. With the growing complexity of satellite data, relational databases are not the best choice for storing satellite data. The need for support for multi-dimensional arrays in satellite data limits the usage of traditional relational databases (Wang et al., 2019; Krčál and Ho, 2015; Baumann, 2001). Relational databases however provide a solution to store for example spatial features like polygons and shapes.

Many publications concerning webGIS solutions used vector-based spatial data like points and polygons as the primary data source in the solution. In raster-based spatial data, the actual data is stored pixel-wise as opposed to vector data where data is saved as points, lines and polygon.

Array database management systems (Array DBMS) solve the problem by providing the necessary features to store multi-dimensional arrays. Systems like SciDB (Cudre-Mauroux et al., 2009) and rasdaman (Baumann et al., 2013) enable queries to multi-dimensional data like satellite data. Both systems have options for importing data from raster files and querying data points from them. Array DBMS enable a more flexible way to process satellite data, because of the array format. It is possible to combine different values from different depths of the array to produce new information about the data (Baumann et al., 2013). OGC has developed a query language called Web Coverage Processing Service (WCPS) that can be used to perform queries to Array DBMS. For example, a query could return the pixel-wise difference of two bands in the satellite image (Baumann, 2010).

For visualizing purposes, the data can be stored in tiles if for example WMTS is used to display results from satellite images. WMTS protocol uses a tiling system to serve spatial data and organizing the data beforehand can increase the efficiency and speed of the WMTS protocol. There are three main options for storing tiled spatial data. Each tile can be stored in separate files and the folder structure is formatted by Tile Matrix, Tile Column and Tile Row. The second methods use also files to store spatial data, but each Tile Matrix has one file where data is stored. Tiled spatial data can also be stored

in relational DBMS where each table row has columns for Tile Matrix, Tile Column, Tile Row and image data. Figure 3.6 shows those three storing methods with incoming WMTS **GetTile** request. In a book by Sample, et al. these three methods are tested, and the method where a single file represents one Tile Matrix or zoom levels was found the quickest solution. The authors also stated that for small datasets of few thousand tiles the single file per tile method is also efficient (Sample and Ioup, 2010).

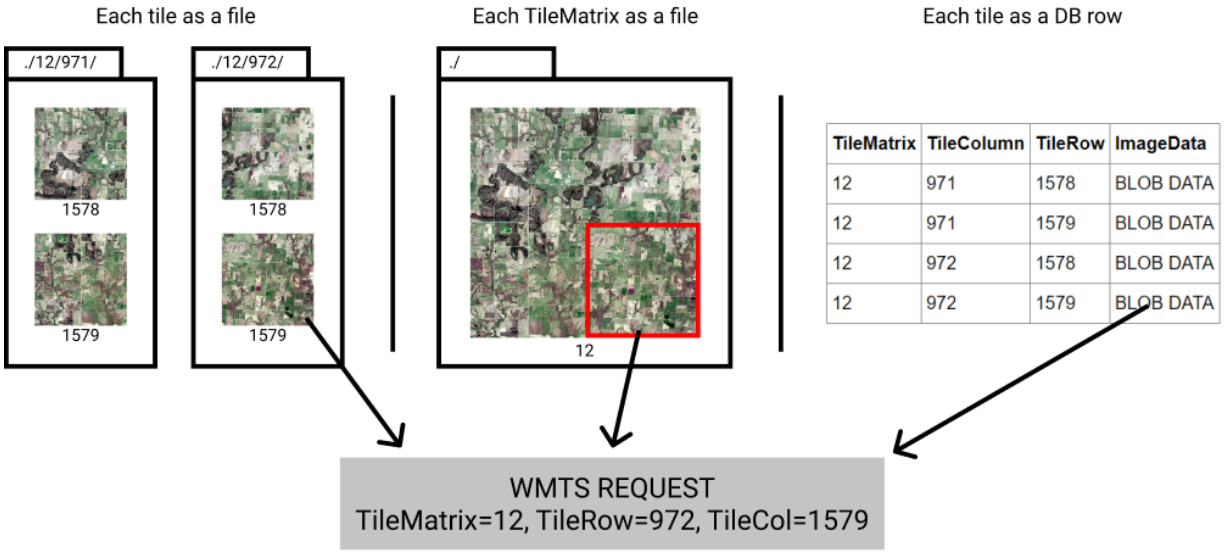

Figure 3.6: Three different ways of storing tiled spatial data visualized (Terramonitor, 2021; Sample and Ioup, 2010, Chapter 7).

Cloud providers have services that can store spatial data for GIS solutions. Cloud provides like AWS offers cloud computing instances that can run the chosen Array DBMS. For storing tiled spatial data AWS offers an efficient key-value storage solution that can be used to host prerendered tiles to be used by WMTS (Sample and Ioup, 2010). The elasticity of cloud instances allows the system to become more scalable which helps with various system loads. Computing instances can be connected to storage services that are responsible for the actual data storage. To increase the transfer speed between the services, the services can be placed in the same virtual cloud network. By doing that the speed between different cloud services is fast.

3.5 Transfering satellite data

Data transfer in satellite image processing has the purpose of connecting the stored satellite data and the client. The type of data transferred to the user depends on the user's need. For example, a client could request tabular data about pixel values or image representation of the data. Data transferred from the client could be new data or instruction for data processing. The field of GIS has many standardized protocols that can be used to transfer data between the client and the GIS solution.

The connection between client and server in GIS solutions can be covered by implementing GIS web services. GIS web services work by providing an endpoint for clients where they can send their requests. These web services can be hosted with existing GIS standards to ensure compatibility with different clients. GIS web services expose only interfaces for connecting server and client, which make the implementation of the standards more efficient and easier (Lu, 2010).

Available coverage standards give clients a way to send instructions for the server to process satellite data. WCPS protocol uses standardized language to send requests to the server for processing. Depending on the storage method of the data the WCPS language must be converted to a suitable query language to retrieve data. Array DBMS solution rasdaman uses its rasql query language to access the stored data. Rasdaman can also process WCPS and WCS request through a PetaScope interface, which is an external service that is capable to convert OGC standardized requests into rasql queries (Aiordăchioaie and Baumann, 2010; Baumann, 2010). Figure 3.7 visualizes a system architecture using rasdaman as Array DBMS and WCPS as access interface to the raster data.

If the coverage is large it is usually observed only small piece at the time. In that case, it is not efficient to transfer all the results back to the client since it will consume a lot of bandwidth and visualizing it will require computing power from the client-side as well. WMTS offer a solution for the visualization problem because it uses tiled data to respond only with the spatial data visible at the time (Sample and Ioup, 2010).

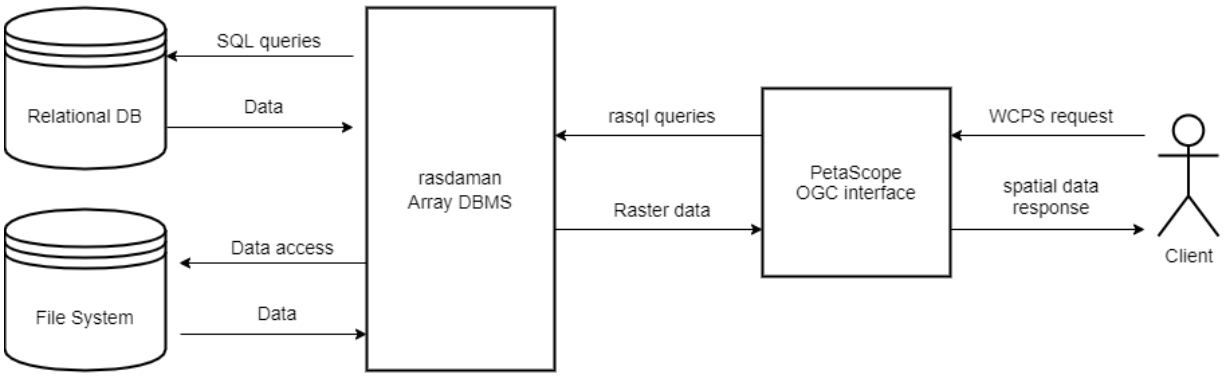

Figure 3.7: Architecture diagram of GIS solution using rasdaman and WCPS initerface. Both supported data storage methods, relational database and file system, are visualized on the left side (Baumann et al., 2013).

3.6 Platforms for Earth Observation Data processing and analyses

Article by R. Antonio et al. introduces a solution that solves the problem of storage and hosting of spatial data. A solution called ChronosDB has the features to visualize spatial data with WMTS protocol after client processing request written in ChronosDB query language. The solution gives an efficient example of data transferring when processing and visualizing spatial data. The overview architecture of the solution contains Array DBMS that is running on computer clusters. ChronosDB then offers a toolkit of raster algorithms that clients can use to perform analysis. After processing the results are server by WMTS server (Ramon Antonio, 2019).

Article by V. Gomes et al. gives an overview of seven GIS solutions that are based on the web and are used to process and analyse Earth Observation Data. The solutions presented in the article implement some of the same features that are needed in the real-time processing platform. Solutions like Google Earth Engine (GEE) and Sentinel Hub (SH) provide high data abstraction which allows users to query data with the tools these platforms are offering. GEE and SH don't provide open-source access to architecture. Open source solutions introduced in the article included solutions like Open Data Cube

(ODC) and OpenEO. ODC combines the storage capability from distributed file systems and cloud providers with OGC's standards to provide API access to clients. ODC handles data loading by breaking it down to smaller blocks that are then accessible via Python API. This solution gives a good approach to data access by providing OGC interfaces like WMTS, but the storage mostly uses file storage rather than Array DBMS. OpenEO does provide the same OGC functionality. But it also supports the usage of Array DBMS solutions like Rasdaman (Gomes et al., 2020).

This section used literature review as a method to find techniques used to store and host spatial data in current GIS solutions. With a broad knowledge about current solutions, it is easier to make architectural decisions on the architectural model. Because of the growing amount of spatial data available the solutions used to process the data has been developing fast in recent years. The need for new DBMS has sped up the development of new Array DBMS systems. Although the storage systems like Array DBMS has become efficient the problem of processing data locally decreased the efficiency of Spatial data processing. Moving Code paradigm introduces a way where the code or algorithm needed to be executed is moved to the server-side. Cloud providers provided a way to build the infrastructure using the Moving Code paradigm. A highly scalable cloud environment combined with standardized web services like WCS and WMTS assures that the GIS solutions work seamlessly with different clients (Gomes et al., 2020). Based on the studied literature efficient web-based GIS solutions should have scalable cloud infrastructure (Ramon Antonio, 2019; Gomes et al., 2020; Riisager et al., 2016), a storage system that supports multidimensional spatial data (Krčál and Ho, 2015; Gomes et al., 2020; Ramon Antonio, 2019) and interfaces that uses standardized web services (Lu, 2010; Gomes et al., 2020; Coimbra, 2009). The findings in this section will be used in section 5 where the architecture model for the real-time processing platform is built. The finding will also help to understand the use-cases examined in section 4.

4 Use cases for real-time processing platform

Terramonitor uses satellite images to provide services for customers in different fields. In some projects, Terramonitor uses pre-processed satellite data that is then used to perform analysis on the given area of interest to meet the customer's needs. The growing number of projects has created a demand for an internal platform that could be used to perform analysis on satellite data. In this section answer to research question two, what are the use cases for the real-time processing platform, is developed by surveying the use cases and requirements for real-time processing platform idea. Terramonitor has completed many projects involving satellite data analysis and the reports for those projects are used to figure out common problems that could be solved with the real-time processing platform. In addition to project reports requirements are gathered from Terramonitor's technical employees. Findings from section 3 are linked to the project reports examined in this section to find out how different technologies are used in the current solution

4.1 Satellite analysis projects

To gather requirements for the real-time processing platform six different projects are inspected to find use cases for the real-time processing platform. Each project is inspected separately and attention is focused on the inputs and outputs of the project. Project input includes the data and data formats that are used as the base data for the project. Project output tells the wanted result from the project. It contains information about the result data and the format and data transfer method of the result data. Table 4.1 lists all six inspected projects, the field of the customer, input data and output results.

Project 1 was completed for a forestry company that was interested in the development of their forest stands. Input data for the project were four-channel satellite images (RGB + NIR). The images covered the forestry stands, which were the areas of interest for this project. There was one image per year for the inspection period. These input images were locally processed and three different visualizations per image were made. For every image a scaled true colour, scaled false colour and NDVI heatmap was generated and they

Table 4.1: Table of inspected projects

Project	Project field	Input data	Output data
1	Forestry	yearly RGB + NIR GeoTIFF images	yearly RGB, RGNIR & NDVI WMTS layers
2	Power grids	RGB + NIR GeoTIFF images	up-to-date RGB & RGNIR layers
3	Power grids	one-band GeoTIFF images	Risk heatmap WMTS layers
4	forest fires	one-band GeoTIFF images	custom coloured heatmap WMTS layer
5	Green buildings	one-band GeoTIFF images	custom coloured heatmap WMTS layer
6	GIS specialists	Sentinel-2 GeoTIFF images & custom masks	cropped NDVI heatmap

were hosted on Terramonitor cloud infrastructure and were delivered for the customer using WMTS protocol. As a result of the project, the customer was able to integrate the WMTS tiled layers into their own GIS solution. Figure 4.1 visualizes the GeoTIFF input and the WMTS output for the project.

The purpose of projects 2 and 3 was to help power grid companies to monitor the surroundings of powerlines. In project 2 four-channel Very High Resolution (VHR) satellite images in GeoTIFF format were acquired from the power line areas. Images were processed and used to produce scaled true colour and scaled false colour WMTS layers. The WMTS layers contained multiple images so that the newest possible image was always on top. Project 3 used one-band satellite images from the same areas to produce risk maps about the surroundings on power line areas. One-band satellite images were analysed, and the result was a heatmap WMTS layer based on the analysed GeoTIFF images. Figure 4.2 shows the input data and results of projects 2 and 3.

The goal of project 4 was to help to prevent forest fires using satellite data. The input data for this project were multiple on-band GeoTIFF images with the date they were captured. The input data was analysed and a severity heatmap was generated based on the analysis. The severity heatmap shows the severity of burnt areas. The heatmap was then coloured and it was provided for the user via WMTS protocol. The layer tiles were stored on AWS storage server and the WMTS service was also running on AWS cloud

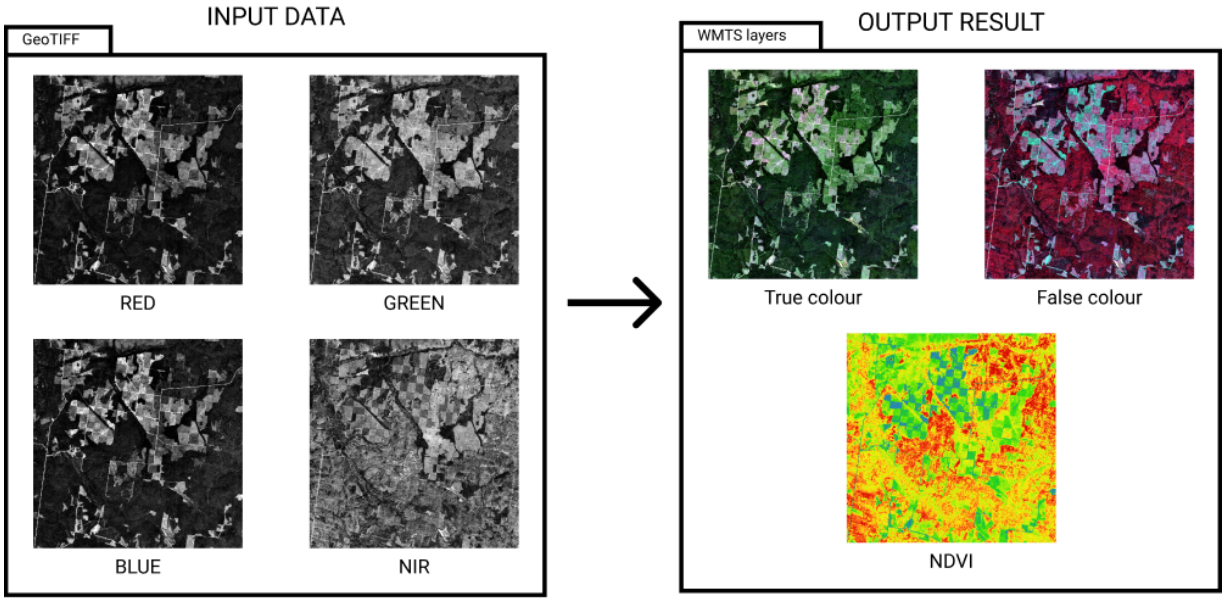

Figure 4.1: The input data for Project 1 and the WMTS layers generated (Terramonitor, 2021).

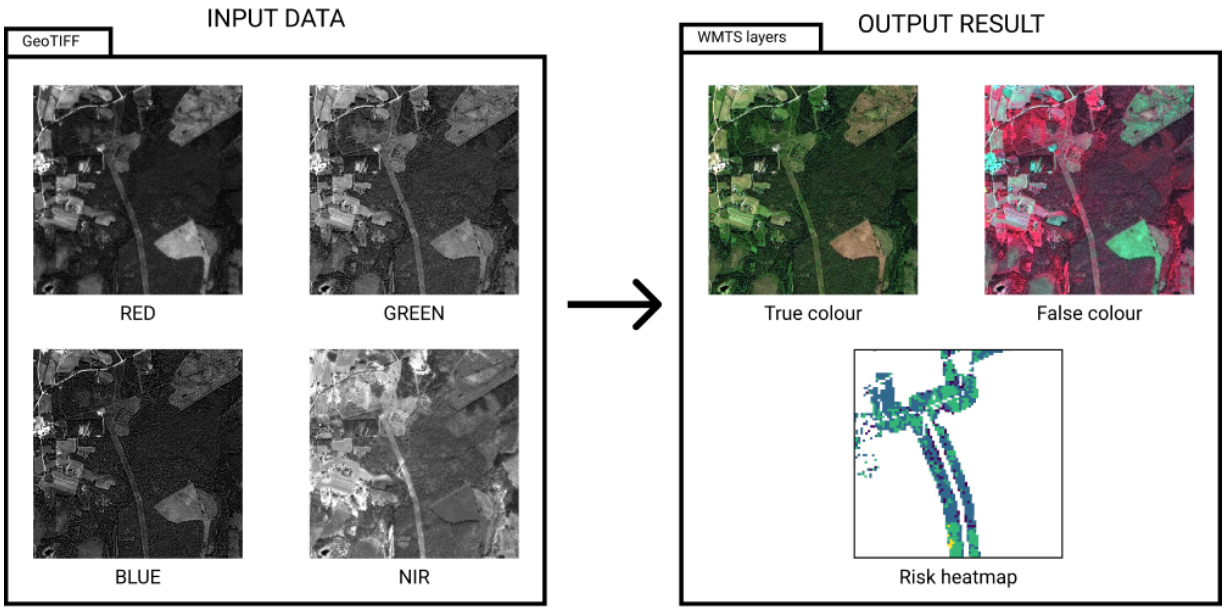

Figure 4.2: The input data for Project 2 and 3 and the WMTS layers generated (Terramonitor, 2021).

computing instance. Figure 4.3 shows the one-band GeoTIFF used and the heatmap layer that was generated.

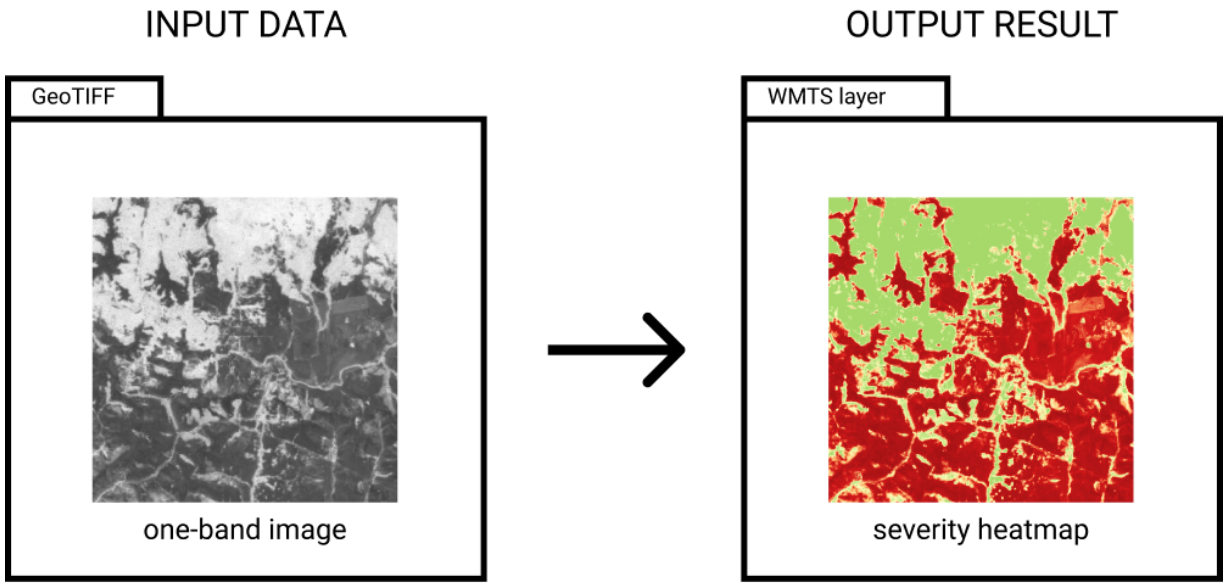

Figure 4.3: The input data for Project 4 and the WMTS layer generated (Terramonitor, 2021).

In project 5 satellite data was used to gather data about the environmental impact of urban areas. For this project, the input data was multiple one-band GeoTIFF images with the known capture time. These GeoTIFF images were then analysed to create a heatmap that could be accessed through a WMTS interface. The WMTS layer was implemented so that the newest data was always on top. AWS was used to store and host the tiled result data. Figure 4.4 visualizes the input and output for the heatmap layer that is using data from the Sentinel-5 satellite to visualize the amount of methane in the atmosphere.

Terramonitor has developed a way to produce pre-processed analysis-ready Sentinel-2 image mosaics that can be used to perform analysis to satellite data. By using that analysis-ready data Terramonitor has developed an index for identifying shrubs in young forest stands. For this project input data was a combination of analysis-ready Sentinel-2 GeoTIFF images and mask images to mask the areas of forest seedlings. The output for this project was an NDVI heatmap that was masked using the seedling masks. Figure 4.5 shows the tiled input data and the WMTS heatmap layer generated.

When looking at the inputs and outputs of these six example projects the input is GeoTIFF files and the output is a WMTS layer. At the moment the process to complete these projects requires a lot of manual work and waiting time when data is transferred. In

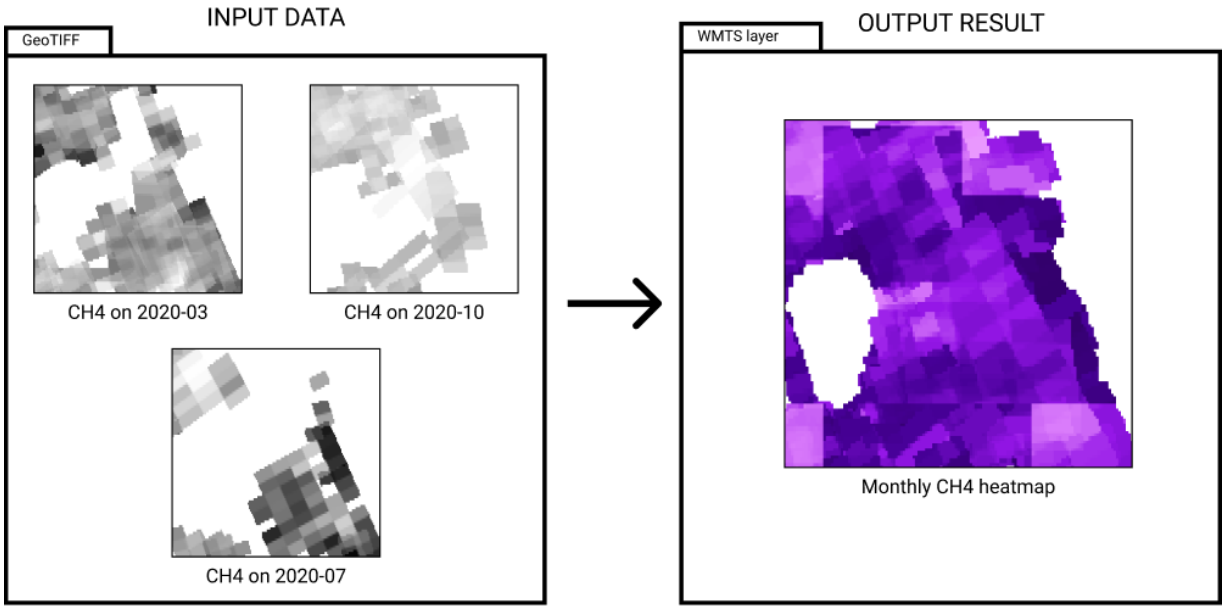

Figure 4.4: The input data for Project 5 captured from Sentinel-5 satellite and the WMTS layer generated (Terramonitor, 2021).

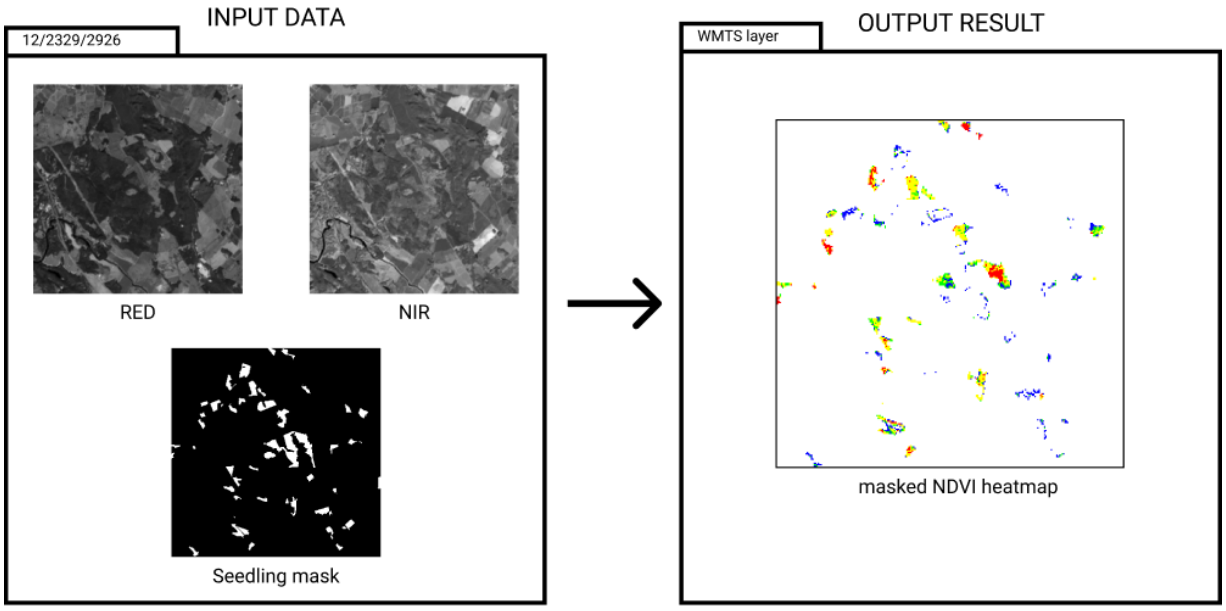

Figure 4.5: The input data for Project 6 and the WMTS layer generated (Terramonitor, 2021).

the current solution, GeoTIFF files are downloaded to a local computer where a GIS specialist performs analysis for the data to produce results. Results are then saved in RGB GeoTIFF files that can be observed by human eyes. To allow clients to request result using WMTS protocol the RGB GeoTIFF files need to be transferred to a server that can convert GeoTIFF files into tiled images. Figure 4.6 shows the current process used to perform analysis to GeoTIFF files and serving results using WMTS protocol.

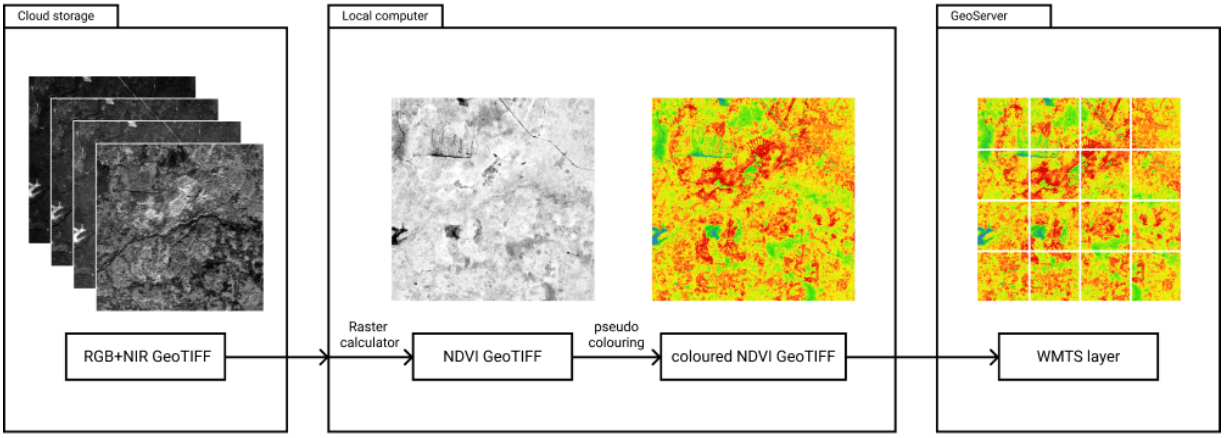

Figure 4.6: Visualization of the process of making NDVI WMTS layer from GeoTIFF image using Terramonitor's current solution (Terramonitor, 2021).

The main requirement for the real-time processing platform is a feature that is capable of converting a GeoTIFF file into tiled images in a cloud environment to serve the GeoTIFF using WMTS protocol. Depending on the purpose of the visualization the GeoTIFF need to go through some analysing processes that require input from the user performing the analysis. In project 1 and 2 scaled true colour and false colour layers were generated. To perform those analyses in the real-time processing platform the platform must have a feature where the user can assign bands from the input image to RGB bands in the output image. In addition to that user must be able to scale the value range in the input image to a range of 0-255 on the output image. With those two features, user can create true colour and false colour images.

In all projects, a WMTS layer like an NDVI layer is produced. To produce custom layers using some index the input image and its band are run through a function that generates the output layer. For these use cases, the real-time processing platform needs to have a feature where the user can define a function for the bands of the input image that is run in

a cloud environment. The resulting image is a one-band image and it needs to be coloured to increase the clarity of the results. To do that users must be able to define colour map and thresholds for the colour map. The user could also decide if the image is coloured using discrete colours or a gradient when colouring the image. In project 6 in addition to creating an NDVI layer, the result is clipped using a mask image. Mask image can define the areas that need to be visible in the resulting image so the real-time processing platform must have a feature where the resulting image is masked using a masking image.

This section examined six different projects to find out the use cases and requirements for real-time processing platform. The found use cases and requirements work as a base for the architecture. With a good knowledge of use cases it is easier to start designing the architecture for a solution that solves the problems defined in the use cases.

5 Architecture for real-time processing platform

The current solution for providing satellite image analysis for Terramonitor's clients require work on local computers that create a bottleneck for a scalable and efficient process. The real-time processing platform can increase the efficiency of the current solution by utilizing cloud environments to do the needed analysis and provide the results for users. Because different cloud services in the cloud environment can form a virtual network, storing and transferring satellite data is secure and very efficient. This section is answering research question three, what architectural decisions need to be made for the real-time processing platform, by using the studied methodology in section three and the gathered requirements in section four.

The architecture presented in this section follows the canonical structure introduced by George Fairbanks. Fairbanks creates a definition for canonical architecture model that includes a set of models that can be used as a framework to create a new architecture. The models in the canonical structure are helping to build the overall architectural model for the system under design (SUD), which is the real-time processing platform. The model has three main models that move from abstract to more concrete ones. These three models are domain, design and code model. The models contain different types of views that visualize the solution from the viewpoint of the view (Fairbanks and Garlan, 2010, Chapter 7). This section is using the definitions created for the Canonical architecture model to present an architectural model that can be used to create the real-time processing platform.

5.1 Microservices Architecture

The architecture for the real-time processing platform will be presented using a microservices architecture. Service-oriented architecture (SOA) divides the solution into smaller services, where each service represents one encapsulated functionality of the solution. The microservice architecture uses similarities from SOA to produce architecture where the application can be run using several independent microservices, that have more autonomy

than the services in SOA (Xiao et al., 2016). Microservice architecture has three bigger benefits when compared to traditional monolith architecture and SOA. The microservice architecture enables fast delivery, efficient scalability and improved autonomy (Jamshidi et al., 2018; Chris Richardson, 2020c).

In Microservice architecture the architecture contains several microservices. Each microservice is responsible for certain features in the application and these microservices are designed to operate independently. Microservice has its own data storage and it communicates with the application and other microservices using messages (Dragoni et al., 2017; Chris Richardson, 2020c). Because microservice is an independent software unit and it only exposes the necessary input and output to other microservices, the microservice is exchangeable and expandable. The concept of independence decreases the delivery time of new features and increases the capability for scaling the application accordingly to the demand (Fowler and Lewis, 2015).

Applications built using microservice architecture are distributed applications, where features are built as independent components and then link to each other accordingly to the requirements for the application. The microservice architecture enables that the different parts of the application can be built using different technologies, languages and programming paradigms (Dragoni et al., 2017).

5.2 Domain model

The domain of real-time processing platform (RPP) covers the need for efficient satellite image analysis and result delivery. A domain model is the most abstract model in the canonical architecture structure (Fairbanks and Garlan, 2010, Chapter 7). The domain model for RPP is created based on the instructions formed by the canonical structure and it includes two different views, a textual information model and a graphical information model. The use cases found in section four are used as a base for these views.

A textual Information model introduced in table 5.1 lists different things and their descriptions related to the domain model of RPP. Things listed in the textual Information model gives an overview of the system and how different factors are related to each other. The textual Information model is supported with a graphical Information model to visualize more precise how different things are related to each other and what kind of links they form in the system (Fairbanks and Garlan, 2010, Chapter 8). Figure 5.1 illustrates the graphical Information model for RPP.

Table 5.1: Textual Information model of real-time processing platform

GeoImage	A raster image containing satellite data
Tile	An image that is part of GeoImage
Layer	A map layer containing tiles generated using RPP
User	A person who has access to RPP and can manage GeoImages, analyses and layers
GeoProject	A project container where project related spatial data can be linked
Analysis	A process where satellite data is processed using a function that manipulates the pixel values
WMTS Interface	An interface that is capable to receive OGC WMTS standard's requests and forward them to other services

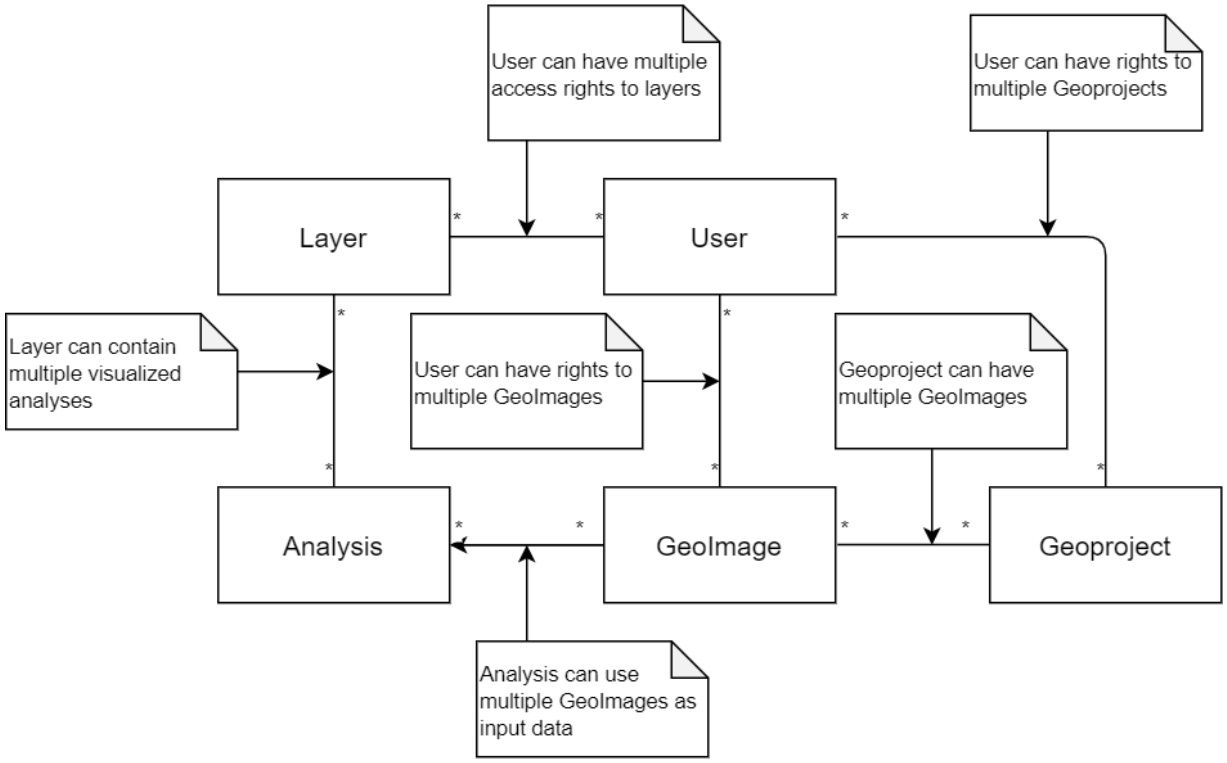

Figure 5.1: Graphical Information model of real-time processing platform.

5.3 Design model

The design model for RPP represents the overview of the architectural structures needed for building the solution. The design model follows the structure of the canonical architecture model, and it also uses design patterns from the microservice style. The design model has views that give a more concrete view on the system than the domain model (Fairbanks and Garlan, 2010, Chapter 8). The views included in the design model of RPP includes a list of use cases and a system context, that shows the interactions between RPP and external systems. Based on use cases and system context the application is divided into subdomain view using core and supporting services. Subdomains are then used to create a component assembly diagram. Main features are also written open as sequence charts to increase the readability of the design model.

5.3.1 Use cases and system context

Use cases are used to list different actions and features that are executable by the users of the solution. Use cases are listed separately for users interacting with satellite data and its analysis and for the user interacting with the results generated using RPP.

The use case diagram for RPP is made using two different users. The first user is a GIS analyst who is a user who is responsible for analysing satellite data and providing the results for customers. Another user for RPP is the customer who can view the results of the analysis or download the result data. Figure 5.5 visualizes the use cases for different actors in RPP.

The system context diagram for RPP is built based on the use cases listed in figure 5.5. System context gives an overview of the actors and the system by connecting the system with external components connected to it (Fairbanks and Garlan, 2010, Chapter 9). In the system context diagram real-time processing platform has three instances where actors can communicate using external services. Figure 5.2 visualizes the system context for RPP.

5.3.2 Subdomains and component assembly

In a microservice architecture, the system can be formed by microservices that are loosely coupled together. The domain of the system can be divided into subdomains using the Domain-Driven Design (DDD) principles. In DDD the domain model is in the centre

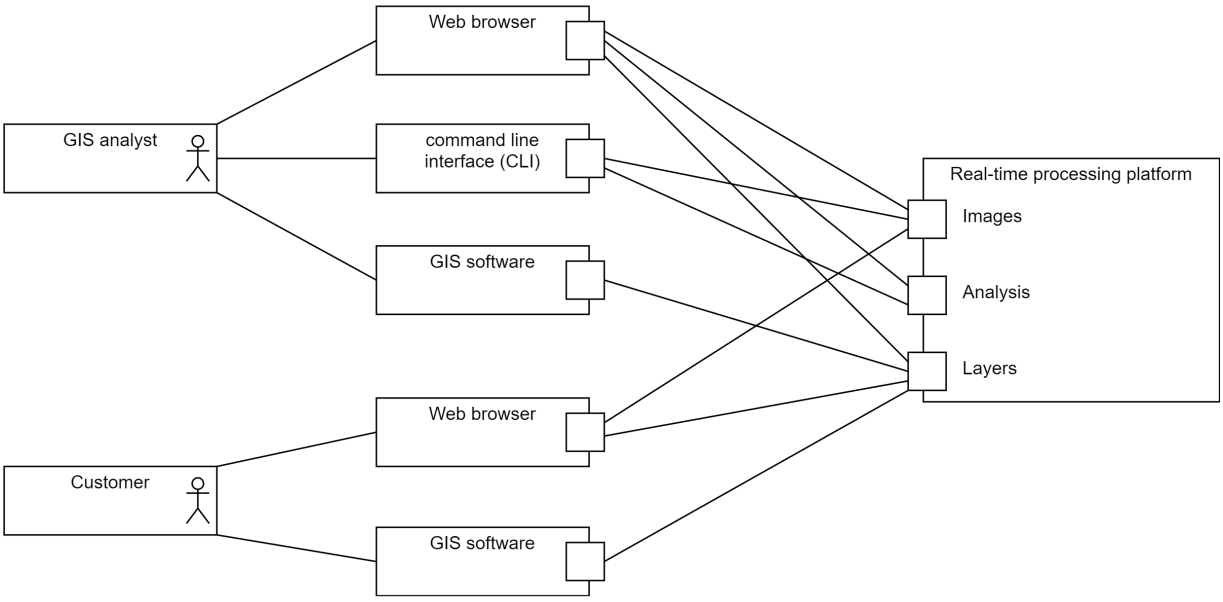

Figure 5.2: System context diagram for real-time processing platform.

of development. DDD states that the domain model has a good view about the system under design and about its features (Martin Fowler, 2020; Fairbanks and Garlan, 2010, Chapter 10). Views on the domain model follow the Single Responsibility Principle (SRP) and by using them as the base for designing microservices, the microservice patterns can be followed easier.

Each subdomain is responsible for one part of the system under design. With the help of a designed subdomain, the system can be divided into microservices that handle the features needed by every subdomain. Subdomains can be divided into core, supporting and generic subdomains. The core contains the subdomains that are the most valuable for the system and executes the main features. Supporting subdomains includes subdomains that are related to the system, but their features can also be outsourced. Generic subdomains help the system but are not tightly related to the system (Chris Richardson, 2020b). Figure 5.3 visualizes the subdomains diagram for the real-time processing platform.

Component assembly gives an overview of the system's internal structure and how internal structures are connected to each other. The component assembly also shows how external services are integrated and how the system interacts with them (Fairbanks and Garlan, 2010, Chapter 9). The component assembly for RPP uses the subdomain diagram to build visualize the needed microservices and their relationships to each other and to external

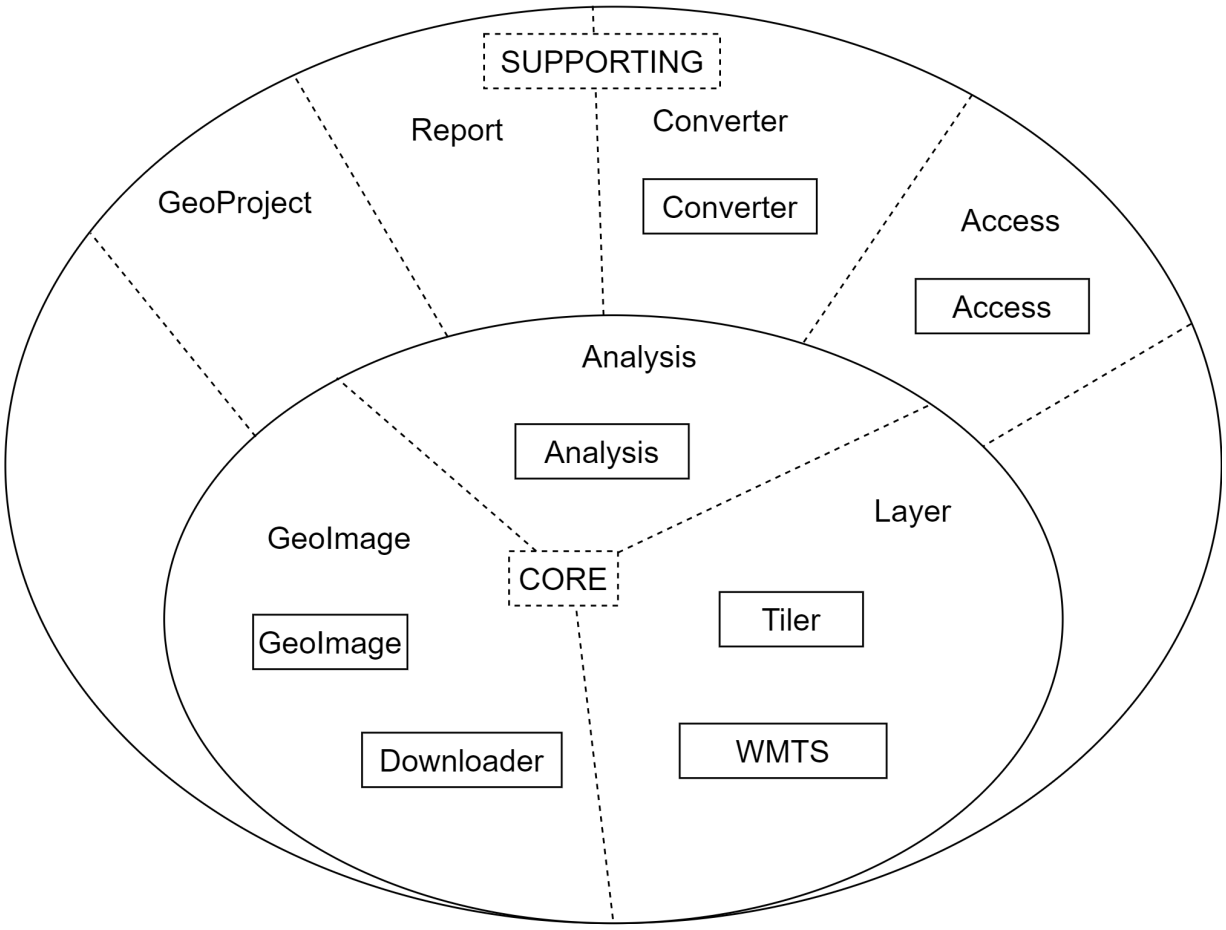

Figure 5.3: Subdomain diagram for real-time processing platform containing core and supporting subdomains.

services. The structure of component assembly follows design patterns from microservice architecture and each component in the diagram represents a microservice. RPP has three different endpoints related to analyses, geo images and layers. At the bottom of the diagram, the needed independent databases or data storages are visualized. Figure 5.4 shows the component assembly diagram of RPP.

5.4 Code model

The third and most concrete model in the canonical architecture model is the code model. The code model represents the source code of the solution. Because the design model shows an abstract view of the architecture there can be a gap between the source code in

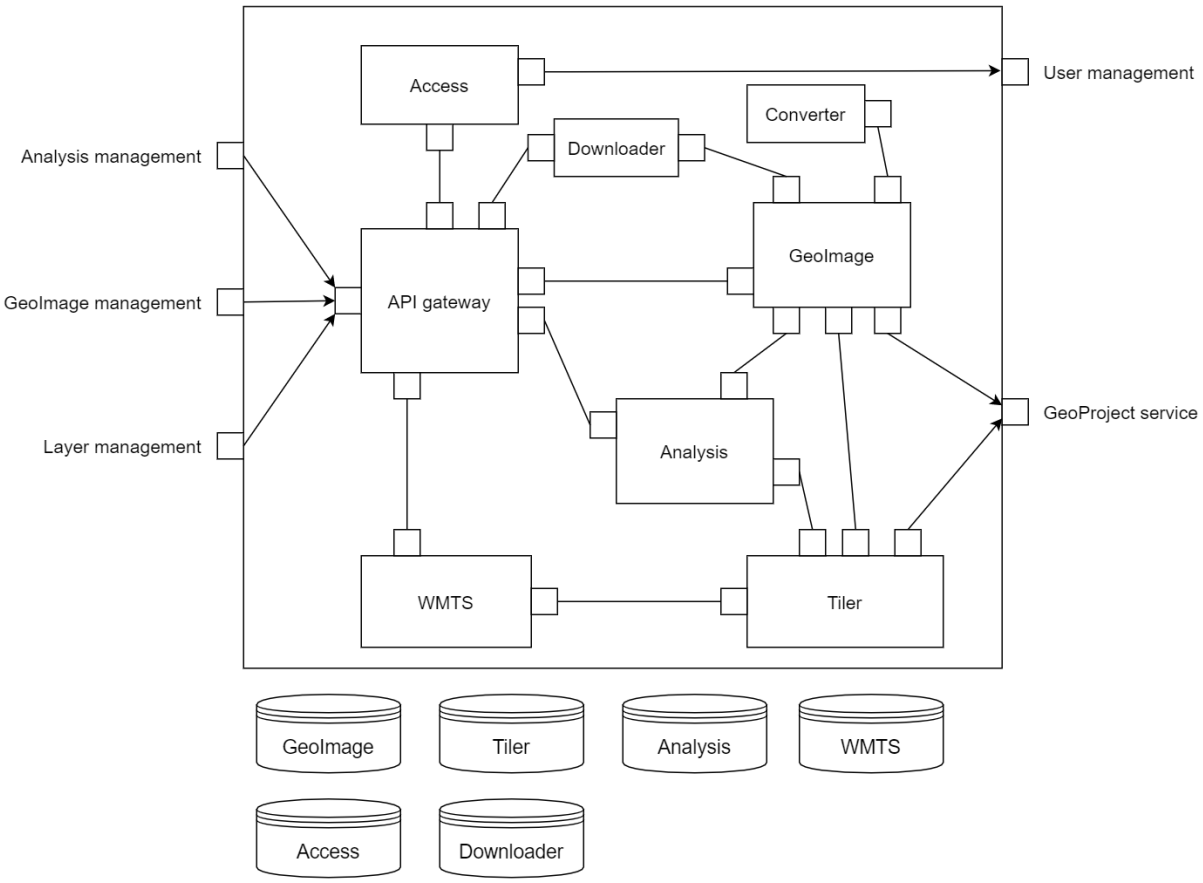

Figure 5.4: Component assembly diagram for real-time processing platform containing the needed microservices and databases.

the code model and the views on the design model. This gap is called the model-code gap and it means the difference in how things in architectural models are presented differently in the source code (Fairbanks and Garlan, 2010, Chapter 10). By understanding the meaning of the model-code gap and the differences between the models, it helps to use the architectural model more effectively in the development process, where the code model is created.

The differences can be found for example, by looking at the vocabulary listed in the design model. Those elements in the design model can be then linked to the programming techniques that are used, and in RPP they can be represented as individual microservices. Because the design model is an abstract view of the system, the microservices presented in it can contain smaller services, like classes or packages, that are running inside the microservice.

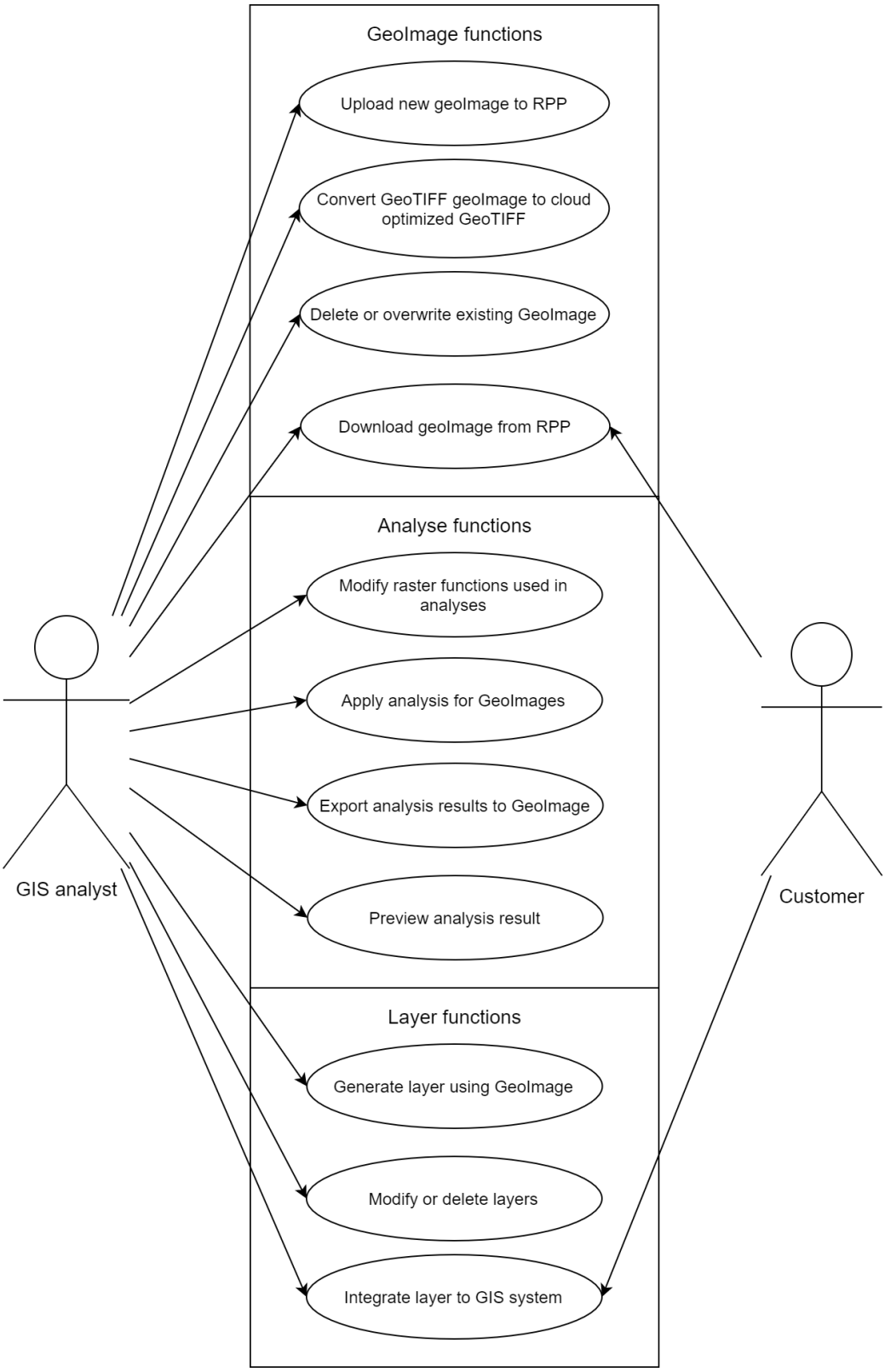

Figure 5.5: Use case diagram for real-time processing platform.

The design model contains both intensional and extensional elements, while the code model includes only extensional. Intensional elements are elements that are quantified across all elements and because they are more like general rules, they are not directly expressed in the source code. Extensional elements like components in the design model can be directly linked to an element in the source code (Fairbanks and Garlan, 2010, Chapter 10).

This division of models from abstract and intensional to extensional and concrete allows more complex architectural models to be built because complex and big systems are easy to present in a simpler and abstract way. In order to produce a more concrete model and the source code, it is important to keep the models synced during the process (Fairbanks and Garlan, 2010, Chapter 10). In RPP the usage of microservice architecture helps to keep models synced because it allows the system to be divided into smaller independent parts.

5.5 Microservices

5.5.1 GeoImage subdomain

The GeoImage subdomain is responsible for handling the storage of satellite images in RPP. This subdomain includes two microservices, GeoImage and Downloader microservices. GeoImage microservice handles the storing of the images and the downloader microservice handles the data transfer to the user if images are downloaded.

The GeoImage microservice is used to handle the incoming data and transferring it to the data storage. Input data format for the RPP is GeoTIFF images because GeoTIFF format supports multi-band images with the data needed for geocoding and georeferencing. The incoming GeoTIFF is first checked if it is cloud optimized. If the image is not cloud optimized the Converter microservice will convert the GeoTIFF to a Cloud Optimized GeoTIFF image. GeoImage microservice has two databases, one array DBMS and one file system. The raw GeoTIFF file is stored into Simple Storage Service (S3) that is a storage service provided by AWS. GeoTIFF is also imported to rasdaman array DBMS where it will be available for other microservices through the GeoImage service.

The Downloader microservice is used to give users an endpoint to download GeoTIFF images from RPP. Downloader microservice receives a list of images that are needed, and it provides a list of locations where those images can be downloaded from. Downloader is a separate microservice, because its lightness doesn't require the possibility to scale up,

and it can be easily replaced in the future.

5.5.2 Analysis subdomain

The Analysis subdomain contains the features needed to generate analyses for satellite images stored by the GeoImage subdomain. Analysis subdomain contains one microservice. Analysis microservice is used to execute analysis and stores the results using GeoImage or Tiler microservices.

The Analysis subdomain has three main functions that can be used to execute analysis to satellite image data in the array DBMS. Raster calculation functions can be used to produce scaled images, like true colour and false colour images, or one-band images, like NDVI using a raster equation. The second function is used to crop a raster image using a masking image. The third function allows the creation of pseudo colouring to a one-band image. All the analyses are stored in a database hosted by the microservice. Results are stored in GeoImage microservice in GeoTIFF format to be used in Downloader microservice. Tiler microservice uses the result GeoTIFF to create a tiled layer that can be requested via the WMTS microservice.

5.5.3 Layer subdomain

The Layer subdomain handles visualization of the analysis in RPP by providing an interface for users that they can use to request map views. The subdomain is responsible for generating the tiles based on the analysis. Users can then request the tiles using the WMTS tiled layers protocol. Layer subdomain contains two microservices that provides the features provided by the subdomain.

Tiler microservice contains all the functionality needed to convert GeoTIFF images into tiles. Tiler receives analysis data from the Analysis microservice and based on the data receives the microservice can access the right GeoTIFF file located in the GeoImage microservice. Tiler divides the GeoTIFF into tiles by storing each TileMatrix into an own file in S3 file storage. Tiler also stores data about the available layers and shares that data with the WMTS microservice.

WMTS microservice is capable of receiving WMTS requests and sending tiles or layer data as a response. WMTS standard is used because it allows easy integration to external GIS systems or web-based map viewers. The microservice provides GetCapabilities and GetTile

endpoints. GetCapabilities endpoint serves a document that includes all the layers that are available for the user. The layer data is fetched from the Tiler microservice. GetTile endpoint can be used to receive tiles from the analysis made by RPP. Tiles are fetched from Tiler microservice's tile storage.

5.5.4 Supporting subdomains

Supporting subdomains are subdomains that contain features that can be outsourced to external services. In RPP most of the supporting subdomains have features that are used in other services that are provided by Terramonitor and therefore they can be used by multiple solutions.

The GeoProject subdomain contains functionality related to geoprojects. Geoproject is used as a container that includes all the needed spatial data, like area polygons and raster images, that are related to an ongoing project. Because geoprojects can be used in other services provided by Terramonitor, the Geoproject service is an external service that RPP uses when it needs to manipulate an individual project's container. In RPP the GeoImage and Tiler microservices communicate with the GeoProject service to add new images or layers into the project container.

Converter subdomain contains a Converter microservice that has the responsibility of executing spatial convert functions for spatial data. In the component assembly diagram the Converter, microservice connects to GeoImage microservice. Converter microservice will convert GeoTIFF images into Cloud Optimized GeoTIFF if needed.

The Access subdomain contains an Access microservice, that verifies that the user has access right for the requested service and its response. Access service connects to the user management of Terramonitor that is an external service. Access service can verify that the user in question has the right to do the action that is requested. After successful authentication Access service gives permission to the API gateway to send the request to the right microservice.

The Reporting subdomain contains functionalities that gather metrics about the status and usage of RPP. This information is gathered using external monitoring services. The data gathered can help to find bottlenecks in the system, that could need refactoring in the future. Reporting subdomain works closely with the cloud platform in order to scale the services to match the demand.

All the incoming requests are received in the API gateway microservice that authenticates

the request and sends it to the right microservice. The usage of API gateway exposes only one endpoint for users and therefore it insulates the inner structure of the solution (Chris Richardson, 2020a). API gateway also supports requests for multiple microservices, which reduces the number of needed requests.

This section created an architectural model for the real-time processing platform using a canonical architecture model and microservice patterns. The architecture was created by first using abstract views of the system and processing those abstract views into more concrete ones. The requirements for the architecture were found from the use cases that were examined in section 4. The findings in section 3 gave knowledge about the technical decisions needed for handling spatial data, and the findings supported the creation of this architectural model. The answer to the third research question contains the domain and design model of RPP, and both views visualize the model by using different types of views. The code model in this section contained instructions on how the code model could be formed in development and how the model-code-gap should be considered.

The architecture model for RPP follows patterns formed by microservice architecture, and it enables the architecture to be more scalable and efficient than a traditional monolith system architecture. With microservices different part of the system work independently and when this architecture is combined with the elasticity of cloud platforms, the result is a highly scalable and available cloud-based processing platform for satellite data. With the architecture created in this thesis, the development of the real-time processing platform could be started and the ready solution would increase the efficiency of satellite data processing.

6 Conclusions

In this thesis, an architectural model for a real-time satellite data processing platform was built. This thesis introduced the basic concepts related to satellite data and satellite image analysis, that are needed to understand the methodology related to satellite image processing. The literature review on this thesis found data about current GIS solutions and techniques used in them. The architecture was built using the use cases that were gathered from previous satellite data related projects.

The first research question covered the problem related to the storing and hosting of spatial data. Storage and hosting were the main issues in the architectural model for RPP because an efficient storage solution can improve the overall speed of the whole solution. By solving the hosting issues the solution can provide an efficient way for users to view the actual results of the analyses. By looking at the answer to these two problems the architectural decisions related to these topics can be made on a reliable basis.

The answer to the first research question was studied by conducting a literature review on the topics. The main finding was concerning OGC that is a consortium maintaining and developing EO standards. OGC was the maintainer for multiple standards that could be used in spatial data storage and hosting. By using those standards as part of the architecture for RPP the external connectivity of the solution was increased. Usage of the standards helps to receive spatial data and sending results for users. The discovery of Array DBMS gave an efficient way to store spatial data inside the RPP. Especially the query language and WPCS integration that RasDaMan offered suited well with the needs of RPP. To increase the scalability and elasticity of the RPP cloud platforms offered all the infrastructural needs of RPP. Efficient storage and hosting of spatial data can be done using OGC standards, Array DBMS and cloud platforms.

The second research question looked at the use cases for real-time processing platform. Finding the right use cases for RPP was important because the right set of use cases help to gather the right requirements for the solution. The requirements for the solution can be deduced from the use cases found, and the gathered requirements help with the building of the architecture.

The process of finding the right use cases for RPP was done by looking at six different projects that were done in Terramonitor. The projects have involved satellite data analyses

and result delivery for clients. On each project, the input data was examined and use cases related to incoming data were found. From the analysis part of each project, few different main use cases were found and requirements were formed based on them. From the result delivery part of the project, two main delivery methods were found. In every project, GeoTIFF files were used as input and GeoTIFF and WMTS were used as a method for result delivery. The analysis methods for the projects could be divided into three main features.

The third research question used the findings of research questions two and three to form an architectural model for RPP. The purpose of an architectural model is to create a base for the development of RPP and with detailed architecture, architectural decisions are not needed to be done in the development phase.

The architecture used canonical architecture models that allowed to start forming the architecture using abstract views of the solution. The presented architecture contains multiple views on the solution containing both abstract and more concrete views. With the design model presented by the canonical architecture model, the solution's domain was decomposed into smaller subdomains that were used to determine the needed microservices. The microservice architecture was selected to ensure that the solution is scalable and future development is easy. The microservice architecture enables this by dividing the solution into smaller individual services. The created architecture uses OGC standards and provides an interface supporting them, to ensure that the integration is possible with other GIS solutions.

The real-time processing platform solves the problem of manual work in satellite data analysis by allowing the analyses to be run in a cloud environment using RPP. The main purpose of automating manual work is to increase efficiency and lower the delivery times when generating new results from satellite data using different types of analyses. RPP centralizes the features related to image, analysis and layer management into one scalable solution that includes multiple microservices handling the requests. RPP uses modern techniques to handle the problems related to spatial data storage and hosting. The real-time processing platform helps Terramonitor to provide better analyses services with efficient satellite data handling with a result delivery that is highly available for different types of clients in different fields.

Bibliography

Aiordăchioaie, A. and Baumann, P. (2010). "PetaScope: An Open-Source Implementation of the OGC WCS Geo Service Standards Suite". In: *Scientific and Statistical Database Management*. Ed. by M. Gertz and B. Ludäscher. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 160–168. DOI: 10.1007/978-3-642-13818-8_13.

Amazon Web Services (2021). *Earth on AWS*. [Accessed 25.3.2021]. URL: https://aws.amazon.com/earth/.

Battersby, S., Finn, M., Usery, E., and Yamamoto, K. (2014). "Implications of Web Mercator and Its Use in Online Mapping". In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 49, pp. 85–101. DOI: 10.3138/carto.49.2.2313.

Baumann, P. (2001). "Web-enabled raster GIS services for large image and map databases". In: *12th International Workshop on Database and Expert Systems Applications*, pp. 870–874. DOI: 10.1109/DEXA.2001.953165.

Baumann, P. (2010). "The OGC web coverage processing service (WCPS) standard". In: *GeoInformatica* 14.4, pp. 447–479. DOI: 10.1007/s10707-009-0087-2.

Baumann, P., Dumitru, A. M., and Merticariu, V. (2013). "The Array Database That Is Not a Database: File Based Array Query Answering in Rasdaman". In: *Advances in Spatial and Temporal Databases*. Ed. by M. A. Nascimento, T. Sellis, R. Cheng, J. Sander, Y. Zheng, H.-P. Kriegel, M. Renz, and C. Sengstock. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 478–483. DOI: 10.1007/978-3-642-40235-7_32.

Chris Richardson (2020a). *API Gateway*. [Accessed 28.5.2021]. URL: https://microservices.io/patterns/apigateway.html.

– (2020b). *Decompose by subdomain Context*. [Accessed 19.5.2021]. URL: https://microservices.io/patterns/decomposition/decompose-by-subdomain.

– (2020c). *Microservice Architecture*. [Accessed 26.5.2021]. URL: https://microservices.io/patterns/microservices.html.

Coimbra, A. R. (2009). "Geographic Data Integration to Support Web GIS Development". In: *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*. MEDES '09. Association for Computing Machinery. DOI: 10.1145/1643823.1643923.

Cudre-Mauroux, P., Kimura, H., Lim, K.-T., Rogers, J., Simakov, R., Soroush, E., Velikhov, P., Wang, D. L., Balazinska, M., Becla, J., DeWitt, D., Heath, B., Maier, D., Madden, S., Patel, J., Stonebraker, M., and Zdonik, S. (2009). "A Demonstration of SciDB: A Science-Oriented DBMS". In: *Proceedings of the VLDB Endowment*. Vol. 2. 2, pp. 1534–1537. DOI: 10.14778/1687553.1687584.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., and Safina, L. (2017). "Microservices: Yesterday, Today, and Tomorrow". In: *Present and Ulterior Software Engineering*. Ed. by M. Mazzara and B. Meyer. Springer International Publishing, pp. 195–216. DOI: 10.1007/978-3-319-67425-4_12.

Durbin, C., Quinn, P., and Shum, D. (2020). *Task 51-Cloud-Optimized Format Study*. Tech. rep. Raytheon Company. URL: https://ntrs.nasa.gov/api/citations/20200001178/downloads/20200001178.pdf.

European Space Agency (2021). *Sentinel-2*. [Accessed 12.2.2021]. URL: https://sentinel.esa.int/web/sentinel/missions/sentinel-2.

Fairbanks, G. and Garlan, D. (2010). *Just Enough Software Architecture: A Risk-driven Approach*. Marshall & Brainerd. ISBN: 978-0-9846181-0-1.

Fowler, M. and Lewis, J. (2015). "Microservices: Nur ein weiteres Konzept in der Softwarearchitektur oder mehr". In: *Objektspektrum* 1.2015, pp. 14–20.

Gomes, V. C. F., Queiroz, G. R., and Ferreira, K. R. (2020). "An Overview of Platforms for Big Earth Observation Data Management and Analysis". In: *Remote Sensing* 12.8. ISSN: 2072-4292. DOI: 10.3390/rs12081253.

Holmes, C. and Rouault, E. (2017). *Cloud Optimized GeoTIFF in depth*. [Accessed 16.3.2021]. URL: https://www.cogeo.org/in-depth.html.

Jamshidi, P., Pahl, C., Mendonça, N. C., Lewis, J., and Tilkov, S. (2018). "Microservices: The Journey So Far and Challenges Ahead". In: *IEEE Software* 35.3, pp. 24–35. DOI: 10.1109/MS.2018.2141039.

Krčál, L. and Ho, S.-S. (2015). "A SciDB-based Framework for Efficient Satellite Data Storage and Query based on Dynamic Atmospheric Event Trajectory". In: BigSpatial'15, pp. 7–14. DOI: 10.1145/2835185.2835190.

Kwang-Soo Kim, Min-Soo Kim, and Kiwon Lee (1997). "On integrated scheme for vector/raster-based GIS's utilization". In: *IGARSS'97. 1997 IEEE International Geoscience and Remote Sensing Symposium Proceedings. Remote Sensing - A Scientific Vision for Sustainable Development*. Vol. 1, 200–203 vol.1. DOI: 10.1109/IGARSS.1997.615838.

Li, S., Dragicevic, S., Castro, F. A., Sester, M., Winter, S., Coltekin, A., Pettit, C., Jiang, B., Haworth, J., Stein, A., and Cheng, T. (2016). "Geospatial big data han-

dling theory and methods: A review and research challenges". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 115. Theme issue 'State-of-the-art in photogrammetry, remote sensing and spatial information science', pp. 119–133. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2015.10.012.

Liang, S., Li, X., and Wang, J. (2020). *Advanced Remote Sensing 2nd Edition*. Boston: Academic Press. ISBN: 978-0-12-815826-5.

Lu, X. (2010). "An Approach to Service and Cloud Computing Oriented Web GIS Application". In: *2010 International Conference on Internet Technology and Applications*, pp. 1–4. DOI: 10.1109/ITAPP.2010.5566578.

Martin Fowler (2020). *DomainDrivenDesign*. [Accessed 19.5.2021]. URL: https://martinfowler.com/bliki/DomainDrivenDesign.html.

Mell, P. M. and Grance, T. (2011). *The NIST definition of cloud computing*. Tech. rep. Sp 800-145. National Institute of Standards & Technology.

National Land Survey of Finland (2021). *NLS orthophotos*. [Accessed 18.3.2021]. URL: https://www.maanmittauslaitos.fi/en/maps-and-spatial-data/expert-users/product-descriptions/orthophotos.

Open Geospatial Consortium (2009). *Web Coverage Processing Service*. [Accessed 1.4.2021]. URL: https://www.ogc.org/standards/wcps.

– (2018). *Web Coverage Service*. [Accessed 1.4.2021]. URL: https://www.ogc.org/standards/wcs.

– (2019a). *OGC GeoTIFF Standard*. [Accessed 11.3.2021]. URL: https://www.ogc.org/standards/geotiff.

– (2019b). *OGC Two Dimensional Tile Matrix Set*. [Accessed 16.3.2021]. URL: https://www.ogc.org/standards/tms.

– (2019c). *OpenGIS Web Map Tile Service Implementation Standard*. [Accessed 18.3.2021]. URL: https://www.ogc.org/standards/wmts.

– (2021). *Open Geospatial Consortium*. [Accessed 19.2.2021]. URL: https://www.ogc.org/.

OpenStreetMap contributors and Geofabrik GmbH (2021). *Planet dump retrieved from: https://download.geofabrik.de*. [Accessed 8.4.2021].

Ramon Antonio, R. Z. (2019). "ChronosDB in Action: Manage, Process, and Visualize Big Geospatial Arrays in the Cloud". In: *Proceedings of the 2019 International Conference on Management of Data*. SIGMOD '19. Amsterdam, Netherlands: Association for Computing Machinery, pp. 1985–1988. ISBN: 9781450356435. DOI: 10.1145/3299869.3320242.

Riisager, P., Herzberg, P., Bødtkjer, M., Jørgensen, J., Bergstedt, T., and Helsted, C. (2016). "Geodata in the Cloud". In: *Geoforum Perspektiv* 14.26. DOI: 10.5278/ojs.perspektiv.v15i29.1338.

Ritter, N. and Ruth, M. (1997). "The GeoTiff data interchange standard for raster geographic images". In: *International Journal of Remote Sensing* 18.7, pp. 1637–1647. DOI: 10.1080/014311697218340.

Sample, J. T. and Ioup, E. (2010). "Tile Storage". In: *Tile-Based Geospatial Information Systems: Principles and Practices*. Boston, MA: Springer US, pp. 117–131. ISBN: 978-1-4419-7631-4. DOI: 10.1007/978-1-4419-7631-4_7.

Saraswat, M. and Tripathi, R. C. (2020). "Cloud Computing: Analysis of Top 5 CSPs in SaaS, PaaS and IaaS Platforms". In: *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*, pp. 300–305. DOI: 10.1109/SMART50582.2020.9337157.

Simonis, I. (2019). "OGC Standardization: From Early Ideas to Adopted Standards". In: *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pp. 4511–4514. DOI: 10.1109/IGARSS.2019.8899858.

Terramonitor (2021). *Terramonitor*. [Accessed 28.1.2021]. URL: https://www.terramonitor.com/.

Trakas, A. and McKee, L. (2011). "OGC standards and the space community — Processes, application and value". In: *2011 2nd International Conference on Space Technology*, pp. 1–5. DOI: 10.1109/ICSpT.2011.6064683.

Wang, S., Zhong, Y., and Wang, E. (2019). "An integrated GIS platform architecture for spatiotemporal big data". In: *Future Generation Computer Systems* 94, pp. 160–172. ISSN: 0167-739X. DOI: 10.1016/j.future.2018.10.034.

Wiggins, R. H., Davidson, H. C., Harnsberger, H. R., Lauman, J. R., and Goede, P. A. (2001). "Image File Formats: Past, Present, and Future". In: *RadioGraphics* 21.3, pp. 789–798. DOI: 10.1148/radiographics.21.3.g01ma25789.

Xiao, Z., Wijegunaratne, I., and Qiang, X. (2016). "Reflections on SOA and Microservices". In: *2016 4th International Conference on Enterprise Systems (ES)*, pp. 60–67. DOI: 10.1109/ES.2016.14.

Xu, H. and Li, B. (2013). "A study of pricing for cloud resources". English. In: *ACM SIGMETRICS Performance Evaluation Review* 40.4, pp. 3–12.